Naval Research Laboratory
Washington, DC 20375-5320

NRL/MR/5520--94-7442

AD-A277 362

(2)

Design, Development, and Testing of a Network Frequency Selection Service (NFSS)

DENNIS J. BAKER

Communications Systems Branch
Information Technology Division

JASTEJ DHINGRA

Science Applications International Corporation
Arlington, Virginia

DTIC
ELECTE
MAR 2 4 1994
S F D

February 14, 1994

94-09175

Approved for public release; distribution unlimited.

94 3 23 050

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 14, 1994	Final

4. TITLE AND SUBTITLE

Design, Development, and Testing of a Network Frequency Selection Service (NFSS)

5. FUNDING NUMBERS

PE - 62232N
TA - RC32W11
WUAS - DN159-110

6. AUTHOR(S)

Dennis J. Baker and Jastej Dhingra*

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/5520--94-7442

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

NCCOSC-NRaD
San Diego, CA 92152

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

*Science Applications International Corporation, Arlington, VA.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This report focuses on the design, development, and testing of a prototype Network Frequency Selection Service (NFSS) that would automate the selection of a single, best frequency for operation of a single-channel network. After selecting the operating frequency, the NFSS proceeds to set the client network's transmitter and receiver to the chosen frequency. The NFSS is intended to be used as a scheduled service. During the periods when the NFSS is active, it normally tests a subset of the candidate frequencies, along with the previously chosen best frequency, and chooses the new best frequency from among this subset. Eventually, over several such active periods, all candidate frequencies are tested. The client network operates in its normal manner between activations of the NFSS. In the prototype NFSS, the "best" frequency is the one that minimizes the frequency selection metric, which contains, in order of importance: (1) the number of relay nodes disconnected from the main body of the network, (2) the number of relay nodes needed to form a backbone network, (3) the number of links that the given topology falls short of being a fully connected network, and (4) the index of the frequency in the sequential collection of candidate frequencies.

14. SUBJECT TERMS

Networking
H.F.
High frequency communication
Frequency allocation

15. NUMBER OF PAGES

195

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

CONTENTS

EXECUTIVE SUMMARY E-1

1. INTRODUCTION 1

2. ORGANIZATION OF REPORT 1

3. APPLICABLE DOCUMENTS 2
3.1 Government Documents 2
3.2 Non-Government Documents 2

4. DEFINITIONS 3
4.1 Acronyms and Abbreviations 3
4.2 Identifiers Used to Describe System 5
4.3 Symbols 8
4.4 Terms 11
4.5 Revisions to Section 4 12

5. SYSTEM SPECIFICATION 13
5.1 Scope 13
5.1.1 Identification 13
5.1.2 System Overview 13
5.1.3 Section Overview 13
5.2 Applicable Documents 13
5.3 System Requirements 13
5.3.1 Definitions 13
5.3.1.1 General Description 13
5.3.1.2 Missions 14
5.3.1.3 Operational Concepts 14
5.3.1.4 System Design Constraints/Considerations 14
5.3.1.5 System Diagrams 14
5.3.1.6 Some Notational Conventions 14
5.3.2 Characteristics 16
5.3.2.1 Performance Characteristics 17
5.3.2.1.1 Mode A 17
5.3.2.1.1.1 Network Frequency Selection Service (NFSS) 18
5.3.2.1.1.2 Monitor, Controller, and Archiver Capability (MCA_CAP) 19
5.3.2.1.1.3 Test-Plan Description Capability (TPD_CAP) 19
5.3.2.1.2 Mode B 19
5.3.2.1.2.1 Network Frequency Selection Service (NFSS) 19
5.3.2.1.2.2 Monitor, Controller, and Archiver Capability (MCA_CAP) 19
5.3.2.1.2.3 Internal Network Capability (IN_CAP) 19
5.3.2.1.2.4 Test-Plan Description Capability (TPD_CAP) 19
5.3.2.1.3 Mode C 19
5.3.2.1.3.1 Network Frequency Selection Service (NFSS) 19
5.3.2.1.3.2 Monitor, Controller, and Archiver Capability (MCA_CAP) 20
5.3.2.1.3.3 Internal Network Capability (IN_CAP) 20
5.3.2.1.3.4 Test-Plan Description Capability (TPD_CAP) 20

/ Codes

Dist	Avail and / or Special
A-1	

5.3.2.2 System Capability Relationships 20
5.3.2.3 External Interface Requirement 20
5.3.2.3.1 GPS Interface 20
5.3.2.3.2 Modem Control Interface via NOSC BLC 20
5.3.2.3.3 Modem Data Interface via NOSC BLC 21
5.3.2.3.4 Transmitter Control Interface 21
5.3.2.3.5 Receiver Control Interface 21
5.4 Quality Assurance Provisions 21
5.5 Preparation for Delivery 21
5.6 Notes 21
5.7 Revisions to Section 5 21
6. SYSTEM DESIGN 22
6.1 Scope 22
6.1.1 Identification 22
6.1.2 System Overview 22
6.1.3 Section Overview 22
6.2 Referenced Documents 22
6.3 Operational Concepts 22
6.3.1 Mission 22
6.3.1.1 User Needs 22
6.3.1.2 Primary Mission 23
6.3.2 Operational Environment 23
6.3.3 Support Environment 23
6.3.3.1 Support Concept 23
6.3.3.1.1 Government and Contractor Support 24
6.3.3.2 Support Facilities 24
6.3.3.2.1 CSS Testbed at NCCOSC-NRaD 24
6.3.3.3 Supply System 24
6.3.3.4 Government Agencies 24
6.3.4 System Architecture 25
6.4 System Design 25
6.4.1 HWCI Identification 27
6.4.2 CSCI Identification 27
6.4.2.1 Network Controller (NC) CSCI 27
6.4.2.1.1 Purpose 27
6.4.2.1.2 System Capabilities Addressed by the Network Controller 27
6.4.2.1.3 External System Interfaces Implemented by the NC 27
6.4.2.2 Link Controller (LC) CSCI 27
6.4.2.2.1 Purpose 27
6.4.2.2.2 System Capabilities Addressed by the Link Controller 28
6.4.2.2.3 External System Interfaces Implemented by the LC 28
6.4.2.2.3.1 GPS Interface (IF_LC_GPS) 28
6.4.2.2.3.2 Modem Interface via NOSC BLC (IF_LC_NOSC) 28
6.4.2.2.3.3 Transmitter Control Interface (IF_LC_XCTL) 28
6.4.2.2.3.4 Receiver Control Interface (IF_LC_RCTL) 29
6.4.2.2.4 Design Constraints 29
6.4.2.3 System Controller (SC) CSCI 29
6.4.2.3.1 Purpose 29

6.4.2.3.2 System Capabilities Addressed by the System Controller 29
6.4.2.3.3 External Interfaces 29
6.4.3 Manual Operations Identification 29
6.4.3.1 Describe Test (SET_TEST) 29
6.4.3.2 Start System (SYS_START) 29
6.4.3.3 Monitor and Maintain System (SYS_RUN) 30
6.4.3.4 Shutdown System (SYS_STOP) 30
6.4.4 Internal Interfaces 30
6.4.4.1 HWCI-to HWCI Interfaces 30
6.4.4.2 HWCI-to-CSCI Interfaces 30
6.4.4.3 CSCI-to-CSCI Interfaces 30
6.4.4.3.1 IF_OP_SC 31
6.4.4.3.2 IF_SC_NC 31
6.4.4.3.3 IF_SC_LC 31
6.4.4.3.4 IF_NC_LC 31
6.4.4.3.5 IF_NC_NC 31
6.4.4.3.6 IF_LC_LC 31
6.5 Processing Resources 31
6.5.1 System Console (CONSOLE) 31
6.5.2 Non-Real-time Network Controller Processor (NC_NRT) 32
6.5.3 Real-time Network Controller Processor (NC_RT) 32
6.5.4 Shared Memory (M224NRL) 32
6.5.5 Ethernet Controller (LANCNTR) 32
6.5.6 Link Controller Processor (M333NRL) 32
6.6 Quality Factor Compliance 32
6.7 Requirements Traceability 32
6.8 Notes 33
6.9 Revisions to Section 6 33
7. SOFTWARE DEVELOPMENT PLAN 34
7.1 Scope 34
7.1.1 Identification 34
7.1.2 System Overview 34
7.1.3 Section Overview 34
7.2 Referenced Documents 34
7.3 Software Development Management 34
7.3.1 Project Organization and Resources 34
7.3.1.1 NRL Facilities to be used 34
7.3.1.2 Government Furnished Equipment, Software, and Services 34
7.3.1.3 Organization Structure 35
7.3.1.4 Personnel 36
7.3.2 Schedule and Milestones 36
7.3.2.1 Activities 37
7.3.2.2 Activity Network 38
7.3.2.3 Source Identification 38
7.3.3 Risk Management 40
7.3.3.1 Software Language Development 40
7.3.3.2 Multi-Contractor and Multi-Site 41
7.3.3.3 Field-Test Resources Availability 41
7.3.3.4 Minimizing the Risks Associated with Field Tests 41

7.3.4 Security 41
7.3.5 Interface with Associated Contractors 41
7.3.6 Interface with Software IV&V Agents 42
7.3.7 Subcontractor Management 42
7.3.8 Formal Reviews 42
7.3.9 Software Development Library 42
7.3.10 Corrective Actions Process 42
7.3.11 Problem/Change Report 42
7.4 Software Engineering 42
7.4.1 Organization and Resources - Software Engineering 42
7.4.1.1 Organizational Structure - Software Engineering 42
7.4.1.2 Personnel - Software Engineering 43
7.4.1.3 Software Engineering Environment 43
7.4.1.3.1 Software Items 43
7.4.1.3.2 Hardware and Firmware Items 43
7.4.1.3.3 Proprietary Nature and Government Rights 43
7.4.1.3.4 Installation, Control, and Maintenance 43
7.4.2 Software Standards and Procedures 44
7.4.2.1 Software Development Techniques and Methodologies 44
7.4.2.2 Software Development Files 44
7.4.2.3 Design Standards 44
7.4.2.4 Coding Standards 44
7.4.3 Non-developmental Software 44
7.5 Formal Qualification Testing 44
7.5.1 Organization and Resources - Formal Qualification Testing 44
7.5.2 Test Approach/Philosophy 44
7.5.3 Test Planning Assumptions and Constraints 44
7.6 Software Product Evaluations 45
7.6.1 Organizational Structure - Software Product Evaluations 45
7.6.2 Software Product Evaluations Procedures and Tools 45
7.7 Software Configuration Management 45
7.8 Other Software Development Functions 45
7.9 Notes 45
7.10 Revisions to Section 7 45
8. SOFTWARE REQUIREMENTS 46
8.1 Network Controller (NC) CSCI 46
8.1.1 Scope 46
8.1.1.1 Identification 46
8.1.1.2 NC Overview 46
8.1.1.3 Subsection Overview 46
8.1.2 Applicable Documents 46
8.1.3 Engineering Requirements 46
8.1.3.1 NC External Interface Requirements 46
8.1.3.2 NC Capability Requirements 46
8.1.3.2.1 NC-to-NC Communication (NC_COM) Requirement 46
8.1.3.2.2 Network Frequency Selection Service (NFSS) Requirement 47
8.1.3.3 NC Internal Interfaces 48
8.1.3.3.1 Internet Protocol (IP) Interface (IF_IP) 48

8.1.3.3.2 NFSS to NC_COM Interface (IF_NC_COM_NFSS) 48
8.1.4 Qualification Requirements 48
8.1.5 Preparation for Delivery 48
8.1.6 Notes 48
8.2 Link Controller (LC) CSCI 49
8.2.1 Scope 49
8.2.1.1 Identification 49
8.2.1.2 LC Overview 49
8.2.1.3 Subsection Overview 49
8.2.2 Applicable Documents 49
8.2.3 Engineering Requirements 49
8.2.3.1 LC External Interface Requirements 49
8.2.3.2 LC Capability Requirements 49
8.2.3.2.1 LC-to-LC Communication (LC_COM) Requirement 49
8.2.3.2.2 Node Timing (ND_TIME) Requirement 50
8.2.3.3 LC Internal Interfaces 50
8.2.4 Qualification Requirements 51
8.2.5 Preparation for Delivery 51
8.2.6 Notes 51
8.3 System Controller (SC) CSCI 51
8.3.1 Scope 51
8.3.1.1 Identification 51
8.3.1.2 CSCI Overview 51
8.3.1.3 Subsection Overview 51
8.3.2 Applicable Documents 51
8.3.3 Engineering Requirements 52
8.3.3.1 SC External Interface Requirements 52
8.3.3.2 SC Capability Requirements 52
8.3.3.2.1 System Performance Archiving (SC_ARCHIVE) Requirement 52
8.3.3.2.2 System Monitor and Control (SC_MON) Requirement 52
8.3.3.3 SC Internal Interfaces 52
8.3.4 Qualification Requirements 52
8.3.5 Preparation for Delivery 53
8.3.6 Notes 53
8.4 Revisions to section 8 53
9. INTERFACE REQUIREMENTS SPECIFICATION 54
9.1 Scope 54
9.1.1 Identification 54
9.1.2 System Overview 54
9.1.3 Section Overview 54
9.2 Applicable Documents 54
9.3 Interface Specification 54
9.3.1 Interface Diagrams 54
9.3.2 Network Controller/Link Controller Interface (IF_NC_LC) 56
9.3.2.1 Interface Requirements 56
9.3.2.2 Data Requirements 56
9.3.3 Operator/System Controller Interface (IF_OP_SC) 56
9.3.3.1 Interface Requirements 56
9.3.3.2 Interface Commands 56

9.3.3.2.1 VxWorks Shell Commands 56
9.3.3.2.2 Unix Commands 57
9.3.3.2.3 MVME333Bug Commands 57
9.3.4 System Controller/Link Controller Interface (IF_SC_LC) 57
9.3.5 System Controller/Network Controller Interface (IF_SC_NC) 58
9.3.6 NC-to-NC Peer Interface (IF_NC_NC) 58
9.3.6.1 Interface Requirements 58
9.3.6.2 Data Requirements 58
9.3.7 LC-to-LC Peer Interface (IF_LC_LC) 58
9.4 Quality Assurance 58
9.5 Preparation for Delivery 58
9.6 Notes 58
9.7 Revisions to Section 9 58
10. SOFTWARE DESIGN 59
10.1 Software Design for the Network Controller (NC) CSCI 59
10.1.1 Scope 59
10.1.1.1 Identification 59
10.1.1.2 System Overview 59
10.1.1.3 Subsection Overview 59
10.1.2 Referenced Documents 59
10.1.3 Preliminary Design 59
10.1.3.1 NC Overview 59
10.1.3.1.1 NC Architecture 59
10.1.3.2 NC Design Description 61
10.1.3.2.1 Network Connectivity Learning (NC_NCL) CSC 61
10.1.3.2.2 Network Structuring(NC_NS) CSC 61
10.1.3.2.3 Transmitter Control (NC_XC) CSC 61
10.1.3.2.4 Receiver Control (NC_RC) CSC 62
10.1.3.2.5 Transmit Buffers Management (NC_TBM) CSC 62
10.1.3.2.6 Receive Buffers Management (NC_RBM) CSC 62
10.1.3.2.7 Congestion Control (NC_CC) CSC 62
10.1.3.2.8 Message Router (NC_MR) CSC 63
10.1.3.2.9 Network Manager (NC_NM) CSC 63
10.1.3.2.10 Network Frequency Selection Service (NC_NFSS) CSC 63
10.1.3.2.11 NC Timing Control (NC_TC) CSC 63
10.1.3.2.12 NC_RT System Startup (NC_RT_SU) CSC 63
10.1.3.2.13 NC_NRT System Startup (NC_NRT_SU) CSC 63
10.1.3.2.14 IP Connection (NC_IP_CONN) CSC 64
10.1.3.2.15 LC Connection (NC_LC_CONN) CSC 64
10.1.3.2.16 Real-time Output Server (NC_RTO_SV) CSC 64
10.1.3.2.17 NC Sim++ Interface (IF_SIM) CSC 64
10.1.3.2.18 NC DSPT Interface (IF_DSPT) CSC 64
10.1.3.2.19 Traffic Source/Sink (NC_SRC_SNK) CSC 64
10.1.4 Detailed Design 65
10.1.4.1 Network Connectivity Learning (NC_NCL) CSC 65
10.1.4.1.1 Periodic Probing Algorithm (PPA) CSU 65
10.1.4.1.1.1 Periodic Probing Algorithm Design Specification/Constraints 65
10.1.4.1.1.2 Periodic Probing Algorithm Design 65

10.1.4.1.2 Network Connectivity Learning Algorithm (NCLA) CSU 67
10.1.4.1.2.1 Network Connectivity Learning Algorithm Design Specification/Constraints .. 67
10.1.4.1.2.2 Network Connectivity Learning Algorithm Design 67
10.1.4.2 Network Structuring (NC_NS) CSC 70
10.1.4.2.1 Linked Cluster Algorithm (LCA) CSU 70
10.1.4.2.1.1 Linked Cluster Algorithm Design Specification/Constraints 70
10.1.4.2.1.2 Linked Cluster Algorithm Design 70
10.1.4.3 Transmitter Control (XC) CSC 72
10.1.4.3.1 Transmitter Control Driver (TCD) CSU 72
10.1.4.3.1.1 Transmitter Control Driver Design Specification/Constraints 72
10.1.4.3.1.2 Transmitter Control Driver Design 72
10.1.4.3.2 Transmission Scheduling Algorithm (TSA) CSU 72
10.1.4.3.2.1 Transmission Scheduling Algorithm Design Specification/Constraints 72
10.1.4.3.2.2 Transmission Scheduling Algorithm Design 72
10.1.4.4 Receiver Control (RC) CSC 73
10.1.4.4.1 Receiver Control Driver (RCD) CSU 73
10.1.4.4.1.1 Receiver Control Driver Design Specification/Constraints 73
10.1.4.4.1.2 Receiver Control Driver Design 73
10.1.4.5 Transmit Buffers Management (TBM) CSC 73
10.1.4.5.1 Transmit Buffers Management Algorithm (TBMA) CSU 73
10.1.4.5.1.1 Transmit Buffers Management Algorithm Design Specification/Constraints 73
10.1.4.5.1.2 Transmit Buffers Management Algorithm Design 73
10.1.4.6 Receive Buffers Management (RBM) CSC 74
10.1.4.6.1 Receive Buffers Management Algorithm (RBMA) CSU 74
10.1.4.6.1.1 Receive Buffers Management Algorithm Design Specification/Constraints 74
10.1.4.6.1.2 Receive Buffers Management Algorithm Design 74
10.1.4.7 Congestion Control (CC) CSC 77
10.1.4.7.1 Congestion Control Algorithm (CCA) CSU 77
10.1.4.7.1.1 Congestion Control Algorithm Design Specification/Constraints 77
10.1.4.7.1.2 Congestion Control Algorithm Design 77
10.1.4.8 Message Router (MR) CSC 82
10.1.4.8.1 Broadcast Traffic Routing Algorithm (BTRA) CSU 82
10.1.4.8.1.1 Broadcast Traffic Routing Algorithm Design Specification/Constraints 82
10.1.4.8.1.2 Broadcast Traffic Routing Algorithm Design 82
10.1.4.8.2 Point-to-Point Routing Algorithm (P2PRA) CSU 84
10.1.4.8.2.1 Point-to-Point Routing Algorithm Design Specification/Constraints 84
10.1.4.8.2.2 Point-to-Point Routing Algorithm Design 84
10.1.4.9 Network Manager (NM) CSC 84
10.1.4.9.1 Network Status Monitoring Algorithm (NSMA) CSU 84
10.1.4.9.1.1 Network Status Monitoring Algorithm Design Specification/Constraints 85
10.1.4.9.1.2 Network Status Monitoring Algorithm Design 85
10.1.4.10 Network Frequency Selection Service (NC_NFSS) CSC 86
10.1.4.10.1 Network Frequency Selection Algorithm (NFSA) CSU 86
10.1.4.10.1.1 Network Frequency Selection Algorithm Design Specification/Constraints 86
10.1.4.10.1.2 Network Frequency Selection Algorithm Design 87
10.1.4.11 NC_RT Timing Control (NC_TC) CSC 94
10.1.4.11.1 Timing Controller (TC) CSU 94
10.1.4.11.1.1 Timing Controller Design Specification/Constraints 94
10.1.4.11.1.2 Timing Controller Design 94
10.1.4.12 NC_RT Startup (NC_RT_SU) CSC 94

10.1.4.12.1 NC_RT Main (NC_RT_MAIN) CSU 95
10.1.4.12.1.1 NC_RT Main (NC_RT_MAIN) Design Specification/Constraints 95
10.1.4.12.1.2 NC_RT Main Design 95
10.1.4.13 NC_NRT Startup (NC_NRT_SU) CSC 95
10.1.4.13.1 NC_NRT Main (NC_NRT_MAIN) CSU 95
10.1.4.13.1.1 NC_NRT Main (NC_NRT_MAIN) Design Specification/Constraints 95
10.1.4.13.1.2 NC_NRT Main Design 95
10.1.4.14 IP Connection (NC_IP_CONN) CSC 95
10.1.4.14.1 IP Driver (IP_DRIVER) CSU 95
10.1.4.14.1.1 IP Driver Design Specification/Constraints 95
10.1.4.14.1.2 IP Driver Design 95
10.1.4.15 LC Connection (NC_LC_CONN) CSC 96
10.1.4.15.1 Network Data Distribution System (NDDS) CSU 96
10.1.4.15.1.1 NDDS Design Specification/Constraints 96
10.1.4.16 Real-time Output Server (NC_RTO_SV) CSC 96
10.1.4.16.1 NC Output Server (NC_O_SV) CSU 96
10.1.4.16.1.1 NC Output Server Design Specification/Constraints 96
10.1.4.16.1.2 NC I/O Server Design 96
10.1.4.17 NC Sim++ Interface (IF_SIM) CSC 96
10.1.4.17.1 NC Sim++ (NC_SIM) CSU 96
10.1.4.17.1.1 NC Sim++ Design Specification/Constraints 96
10.1.4.17.1.2 NC Sim++ Design 96
10.1.4.18 DSPT Interface (IF_DSPT) CSC 97
10.1.4.18.1 NC DSPT (NC_DSPT) CSU 97
10.1.4.18.1.1 NC DSPT Design Specification/Constraints 97
10.1.4.18.1.2 NC_DSPT Design 97
10.1.4.19 Traffic Source/Sink (NC_SRC_SNK) CSC 97
10.1.4.20 NC Timing Control (NC_TC) CSC 97
10.1.4.20.1 Timing Controller (TC) CSU 97
10.1.4.20.1.1 Timing Controller Design Specification/Constraints 97
10.1.4.20.1.2 Timing Controller Design 97
10.1.5 NC Data 97
10.1.6 NC Data Files 97
10.1.6.1 Data File to CSC/CSU Cross Reference 98
10.1.6.2 Complan.inp 98
10.1.6.3 Timing.inp 100
10.1.6.4 nc_vx_start.inp 101
10.1.6.5 Message.inp 101
10.1.6.6 Flags.trace 102
10.1.6.7 Nfss.inp 103
10.1.6.8 Log.out 103
10.1.6.9 Stat.out 106
10.1.6.10 Cksum_blk_rcvd.out 106
10.1.6.11 Cksum_blk_good.out 106
10.1.6.12 Msg.trace 106
10.1.7 Requirements Traceability 108
10.1.8 Notes 108
10.1.9 Revisions to Section 10.1 108
10.2 Software Design for the Link Controller (LC) CSCI 108

10.3 Software Design for the System Controller (SC) CSCI 108
10.3.1 Scope 108
10.3.1.1 Identification 108
10.3.1.2 System Overview 108
10.3.1.3 Subsection Overview 108
10.3.2 Referenced Documents 108
10.3.3 Preliminary Design 109
10.3.3.1 SC Overview 109
10.3.3.1.1 SC Architecture 109
10.3.3.2 SC Design Description 109
10.3.3.2.1 SC Startup (SC_SU) CSC 109
10.3.3.2.2 Archiver (SC_ARC) CSC 110
10.3.3.2.3 Monitor (SC_MON) CSC 110
10.3.4 Detailed Design 110
10.3.4.1 SC Startup (SC_SU) CSC 110
10.3.4.1.1 NFSE Startup Scripts (NFSE_StartupScripts) 110
10.3.4.1.1.1 NFSE Startup Scripts Design Specification/Constraints 110
10.3.4.1.1.2 NC_RT.script 111
10.3.4.1.1.3 NC_NRT.script 111
10.3.4.1.1.4 M333NRL.script 111
10.3.4.2 Archiver (SC_ARC) CSC 111
10.3.4.2.1 Archiver Design Specification/Constraints 111
10.3.4.2.2 Archiver Design 111
10.3.4.3 Monitor (SC_MON) CSC 111
10.3.4.4 Monitor Design 111
10.3.5 SC Data 112
10.3.6 SC Data Files 112
10.3.7 Requirements Traceability 112
10.3.8 Notes 112
10.3.8.1 Refinements to the Frequency Selection Metric 112
10.3.8.1.1 Frequency Selection Metric B 112
10.3.9 Revisions to Section 10.3 113

11. INTERFACE DESIGN DOCUMENT 114
11.1 Scope 114
11.1.1 Identification 114
11.1.2 System Overview 114
11.1.3 Section Overview 114
11.2 Referenced Documents 114
11.3 Interface Design 114
11.3.1 Interface Diagrams 114
11.3.2 NC/LC Interface (IF_NC_LC) 115
11.3.3 LC/LC Interface (IF_LC_LC) 118
11.3.3.1 Link-Level Packet (LC_PKT) 118
11.3.4 NC/NC Interface (IF_NC_NC) 119
11.3.4.1 Synchronous Messages 119
11.3.4.1.1 Frame 1 Synchronous Message (SMSG_F1) 119
11.3.4.1.2 Frame 2 Synchronous Messages (SMSG_F2) 120
11.3.4.1.3 Frame 3 Synchronous Messages (SMSG_F3_CH) and (SMSG_F3_NON_CH) 121
11.3.4.2 Asynchronous Messages 122

11.3.4.2.1 Point-to-Point Asynchronous Message (AMSG_P2P) 122
11.3.4.2.2 Broadcast Asynchronous Message (AMSG_BRCST) 123
11.3.4.2.3 Local Asynchronous Message (AMSG_LOCAL) 123
11.3.4.2.4 Null Asynchronous Message (AMSG_NULL) 123
11.3.5 IF_SC_NC 124
11.3.6 IF_SC_LC 124
11.3.7 IF_OP_SC 124
11.4 Notes 124
11.5 Revisions for Section 11 124

12. NFSE IN-LAB TEST PLAN 125
12.1 Scope 125
12.1.1 Identification 125
12.1.2 System Overview 125
12.1.3 Section Overview 125
12.2 Referenced Documents 125
12.3 NFSE In-Lab Test Environment 125
12.3.1 Software Items 126
12.3.1.1 DSPT Software 126
12.3.1.2 Sim++ 126
12.3.1.3 VxWorks 126
12.3.1.4 C++ Compiler 127
12.3.2 Hardware and Firmware Items 127
12.3.3 Proprietary Nature and Government Rights 127
12.3.4 Installation, Testing, and Control 127
12.3.4.1 DSPT Maintenance 127
12.3.4.2 Sim++ Maintenance 127
12.3.4.3 VxWorks Maintenance 127
12.3.4.4 Workstation Maintenance 127
12.3.4.5 Target Hardware Maintenance 127
12.3.4.6 MVME135 Bootrom Maintenance 128
12.3.4.7 MVME333 Bootrom Maintenance 128
12.3.4.8 Support Hardware Maintenance 128
12.4 Formal Qualification Test Identification 128
12.4.1 NFSE System 128
12.4.1.1 General Test Requirements 128
12.4.1.2 Test Classes 128
12.4.1.2.1 Algorithm Tests 128
12.4.1.2.2 Timing and Load Tests 129
12.4.1.2.3 Node Start/Stop/Start Tests 129
12.4.1.2.4 Integration Tests 129
12.4.1.3 Test Levels 129
12.4.1.4 Test Definitions 129
12.4.1.4.1 7-Node Simulation (7NFSE) 129
12.4.1.4.1.1 Test Objective 129
12.4.1.4.1.2 Special Requirements 129
12.4.1.4.1.3 Test Level 129
12.4.1.4.1.4 Test Type or Class 130
12.4.1.4.1.5 Qualification Method 130

12.4.1.4.1.6 Type of Data to be Recorded 130
12.4.1.4.1.7 Assumptions and constraints 130
12.4.1.4.2 Additional Tests 130
12.4.1.5 Test Schedule 130
12.5 Data Recording, Reduction, and Analysis 130
12.6 Notes 132
12.7 Revisions to section 12 132
13. NFSE IN-LAB TEST DESCRIPTIONS 133
13.1 Scope 133
13.1.1 Identification 133
13.1.2 System Overview 133
13.1.3 Section Overview 133
13.2 Referenced Documents 133
13.3 Formal Qualification Test Preparations 133
13.3.1 7-Node Simulation (7NFSE) 133
13.3.1.1 7-Node Simulation Schedule 133
13.3.1.2 7-Node Simulation Pre-Test Procedures 133
13.3.1.2.1 Hardware Preparation 133
13.3.1.2.2 Software Preparation 133
13.3.1.2.3 Other Pre-Test Preparations 141
13.3.2 Additional Tests 141
13.4 Formal Qualifications Test Descriptions 141
13.4.1 7-Node Simulation (7NFSE) 141
13.4.1.1 7-Node Simulation Test 1 (7NFSE-T1) 141
13.4.1.1.1 7-Node Simulation Test 1 Requirements Traceability 141
13.4.1.1.2 7-Node Simulation Test 1 Initialization 142
13.4.1.1.3 7-Node Simulation Test 1 Inputs 142
13.4.1.1.4 7-Node Simulation Test 1 Expected Results 142
13.4.1.1.5 7-Node Simulation Test 1 Criteria for Evaluating Results 144
13.4.1.1.6 7-Node Simulation Test 1 Procedure 144
13.4.1.1.7 7-Node Simulation Test 1 Assumptions and Constraints 144
13.4.1.2 7-Node Simulation Test 2 (7NFSE-T2) 144
13.4.1.2.1 7-Node Simulation Test 2 Requirements Traceability 144
13.4.1.2.2 7-Node Simulation Test 2 Initialization 144
13.4.1.2.3 7-Node Simulation Test 2 Inputs 144
13.4.1.2.4 7-Node Simulation Test 2 Expected Results 145
13.4.1.2.5 7 Node Simulation Test 2 Criteria for Evaluating Results 145
13.4.1.2.6 7 Node Simulation Test 2 Procedure 145
13.4.1.2.7 7 Node Simulation Test 2 Assumptions and Constraints 145
13.4.1.3 7-Node Simulation Test 3 (7NFSE-T3) 145
13.4.1.3.1 7-Node Simulation Test 3 Requirements Traceability 146
13.4.1.3.2 7-Node Simulation Test 3 Initialization 146
13.4.1.3.3 7-Node Simulation Test 3 Inputs 146
13.4.1.3.4 7-Node Simulation Test 3 Expected Results 146
13.4.1.3.5 7 Node Simulation Test 3 Criteria for Evaluating Results 147
13.4.1.3.6 7 Node Simulation Test 3 Procedure 147
13.4.1.3.7 7 Node Simulation Test 3 Assumptions and Constraints 147
13.4.1.4 Other Tests 147
13.5 Notes 147

13.6 Revisions to Section 13 147
14. NFSE IN-LAB TEST REPORTS 149
14.1 Section Overview 149
14.2 Referenced Documents 149
14.3 Test Overview 149
14.3.1 7-Node Simulation (7NFSE) Test Overview 149
14.3.1.1 7-Node Simulation Summary 149
14.3.1.2 7-Node Simulation Record 149
14.4 Test Results 149
14.4.1 7-Node Simulation (7NFSE) Results 149
14.4.1.1 7-Node Simulation Test 1 Results 149
14.4.1.2 7 Node Simulation Test 2 results 157
14.4.1.3 7 Node Simulation Test 3 results 161

LIST OF FIGURES

Figure 1 Node hardware architecture in which the NFSE System is to be embedded. 15
Figure 2 NFSE System context diagram. 16
Figure 3 Modes of NFSE System operation: Mode A - Interleaved operation of NFSS and HAMA/ MINCAP network; Mode B - Interleaved operation of NFSS and internal network: Mode C - Concurrent operation of NFSS and internal network. 17
Figure 4 NFSE System external interfaces. 21
Figure 5 NCCOSC-NRaD CSS Testbed - fixed sites. 24
Figure 6 System architecture showing CSCI's and external interfaces. 25
Figure 7 System timing structure 26
Figure 8 GPS interface. 28
Figure 9 NFSE system internal interfaces. 30
Figure 10 Code 5521's position in NRL organizational structure. 35
Figure 11 NC external interface requirements. 47
Figure 12 NC internal interfaces 48
Figure 13 LC external interface requirements. 50
Figure 14 LC internal interface requirements. 51
Figure 15 SC external interface requirements. 52
Figure 16 NFSE System interfaces 55
Figure 17 Startup display on System Controller screen. 57
Figure 18 NC architecture showing CSCs, key interfaces, and the target hardware. 60
Figure 19 Additional system timing for operation with NFSE System's internal network 61
Figure 20 Pseudocode for the Periodic Probing Algorithm at node i 66
Figure 21 Format of NCLA data packet for a seven node (N=7) network 67
Figure 23 Pseudocode for Procedure handle_connectivity_report from node t. 68
Figure 22 Pseudocode for the NCLA. 69
Figure 24 Pseudocode for the Linked Cluster Algorithm at Node i. 71
Figure 25 Pseudocode for procedure linkup1 at Node i. 71
Figure 26 Pseudocode for procedure linkup2 at Node i. 72
Figure 27 Pseudocode for the TBMA. 75
Figure 28 Outer block pseudocode for RBMA 77
Figure 29 Pseudo code for RBMA asynchronous message handling 78
Figure 30 Format of a Congestion Control data packet containing an updated value for the global congestion control factor 80
Figure 31 Pseudocode for the CCA at node s 81
Figure 32 Pseudocode for Procedure handle_cca_report. 82
Figure 33 Pseudocode for the Broadcast Traffic Routing Algorithm. 83
Figure 34 Format of NFSA data packet for a seven node (N=7) network. 88
Figure 35 Pseudocode for the NFSA protocol for connectivity dissemination. 89
Figure 36 Pseudocode for NFSA procedure handle_connectivity_report from node t. 90

Figure 38 Pseudocode to find the components of an undirected graph. 91
Figure 37 Format of frequency selection Metric. A 91
Figure 39 Pseudocode for finding the backbone network. 92
Figure 40 Pseudocode for the disseminating the "best" metric. 93
Figure 41 Pseudocode for Procedure handle_metric_msg. 94
Figure 42 Sample complan.inp file from node 2. 99
Figure 43 Timing.inp file. 100
Figure 44 nc_vx_start.inp file. 101
Figure 45 flags.trace file used during tests. 102
Figure 46 Sample nfss.inp file. 103
Figure 47 Sample of the data written after each reorganization epoch to the log.out file for node 2. 104
Figure 48 Sample of the data written by the NFSA CSU to the log.out file for node 2. 105
Figure 49 Sample line of output from cksum_blk_rcvd.out at node 2. 106
Figure 50 Sample line of output from cksum_blk_good.out at node 2. 106
Figure 51 Sample output from msg.trace at node 1 107
Figure 52 C language definitions of software module ids. These appear in column 2 of the msg.trace file. 107
Figure 53 C language definitions of message events. These event ids appear in column 3 of the msg.trace file. 107
Figure 54 C language definitions of message types. These types appear in column 5 of the msg.trace file. 108
Figure 55 System Controller (SC) architecture showing CSCs, key interfaces, and the target hardware. 109
Figure 56 Frequency selection Metric B. 112
Figure 57 CSCI interfaces 114
Figure 58 Structure of link-level transmission. 119
Figure 59 Control message for frame 1 transmission from Node i (i=4, N=7) 119
Figure 60 Control message for frame 2 transmission from clusterhead Node i (i=4, N=7) 120
Figure 61 Control message for frame 2 transmission from non-clusterhead Node i (i=4, N=7) 120
Figure 62 Control Message for frame 3 transmission from a clusterhead. 121
Figure 63 Control message for frame 3 transmission from a non-clusterhead 121
Figure 64 Point-to-point message header. 122
Figure 66 Local message header 123
Figure 65 Broadcast message header 123
Figure 67 Null message header 124
Figure 68 Example of file create.dat 140
Figure 69 Example of file nfss.inp 141
Figure 70 Test cycle sequences during a period of increasing noise levels.. 143
Figure 71 Anticipated results for Test 1 based on relative noise levels of the tested channels. 143
Figure 72 Test 1 (metric A) Results: Set 1 151
Figure 73 Test 1 (metric A) Results: Set 2 152
Figure 74 Test 1 (metric A) Results: Set 3 153
Figure 75 Test 1 (metric B) Results: Set 1 154
Figure 76 Test 1 (metric B) Results: Set 2 155

Figure 77 Test 1 (metric B) Results: Set 3 156
Figure 78 Test 2(metric B) Results: Set 1 158
Figure 79 Test 2(metric B) Results: Set 2 159
Figure 80 Test 2(metric B) Results: Set 3 160
Figure 81 Test 3 (metric B) Results: Set 1 162
Figure 82 Test 3 (metric B) Results: Set 2 163
Figure 83 Test 3 (metric B) Results: Set 3 164
Figure 84 Test 3 (metric B) Results: Set 4 165
Figure 85 Test 3 (metric B) Results: Set 5 166
Figure 86 Test 3 (metric B) Results: Set 6 167

LIST OF TABLES

Table 1: Meanings of acronyms and abbreviations 3
Table 2: CSCI Identifiers 5
Table 3: CSC Identifiers 6
Table 4: CSU Identifiers 6
Table 5: Interface Identifiers 7
Table 6: VME-Board Identifiers 7
Table 7: Names of System Capabilities 8
Table 8: Symbols used in this document (except for time-of-occurrence symbols). 8
Table 9: Time-of-occurrence symbols used in this document. 10
Table 10: Software constants. 10
Table 11: Terms defined in this document. 11
Table 12: NFSE System capabilities associated with various system modes. 20
Table 13: Support software for the operational system 23
Table 14: GPS serial time-of-day input format. 28
Table 15: Requirements Traceability Matrix 32
Table 16: Hardware Items to be Obtained 34
Table 17: Software Items to be Obtained 35
Table 18: Schedules and milestones 36
Table 19: Deliverables 36
Table 20: Hardware Resources needed for NFSS Development 39
Table 21: Software Resources needed for NFSS Development 39
Table 22: Hardware Resources needed for Field-Test Node Development at NRL 39
Table 23: Software Resources needed for Field-Test Node Development at NRL 39
Table 24: Hardware Resources needed for final integration with Field-Test System 40
Table 25: Software Resources needed for final integration with Field-Test System 40
Table 26: Assessment of Risk Areas 40
Table 27: Listing of NFSE System interfaces specified in the IRS. 55
Table 28: Hardware Items for the Realtime Testbed 126
Table 29: Frequencies, channel noise levels, and communication ranges for tests 7NFSE-T2. 144
Table 30: Frequencies, channel noise levels, and communication ranges for tests 7NFSE-T3. 146

EXECUTIVE SUMMARY

Background

Multi-hop High Frequency (HF) (2 to 30 MHz) networks are presently being developed as a solution to the problem of communication in the Intra Battle Group (IBG) environment. In the case of networks such as the one proposed in [ES.1], a single frequency is shared among all the nodes in the network. HF communications are subject to many time-varying effects such as Battle Group platform motion, day/night channel effects, changing sea state conditions, frequency versus communication range capability, jamming, and dynamic noise environments. These conditions all contribute to making it difficult to find a single, "best" frequency for network operation. Standardized Automatic Link Establishment (ALE) techniques[ES.2] provide good frequency selection for a single link but they are often slow and do not answer the question of what is the optimum frequency choice for an entire network.

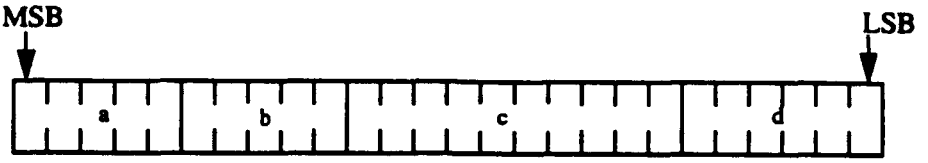
This report focuses on the design, development, and testing of a prototype Network Frequency Selection Service (NFSS) that would automate the selection of a single, best frequency for operation of a single-channel network. For the intended application, the communicating platforms are generally dispersed over an area less than 300 nm across. Communication in this scenario is usually achieved by HF groundwave, although some skywave propagation is possible. Selection of an HF operating frequency for the network could be decided prior to starting up the network and made part of the communications plan. However, problems can arise if the pre-selected operating frequency results in poor communication performance, unless there is some method for switching to a more suitable frequency. The NFSS provides an automated procedure for selecting, from a given sequential collection of HF frequencies, the "best" frequency for use by the network. After selecting the operating frequency, the NFSS proceeds to set the client network's transmitter and receiver to the chosen frequency. The functions of the NFSS are implemented in software to perform this service.

Development of a Prototype of a Network Frequency Selection Service

The NFSS is intended to be used as a scheduled service. That is, it begins and ends operation at predetermined times. During the periods when the NFSS is active it normally tests a subset of the candidate frequencies along with the previously chosen best frequency, and chooses the new best frequency from among this subset. Eventually, over several such active periods, all candidate frequencies are tested. The client network operates in its normal manner between activations of the NFSS.

In the prototype NFSS, the "best" frequency is the one that minimizes the frequency selection metric shown in figure ES-1 (Metric B). The metric is shown here in the form of an integer expressed in binary format. The most significant bit is on the left. The "best" frequency is the one that minimizes this integer. Thus, the most important factor in this metric is the number of unconnected nodes, n_{unc} nodes that are disconnected from the largest component of the network. This implementation attempts to choose the frequency that leaves the fewest nodes disconnected from the largest component of the network. If several frequencies result in the same number of unconnected nodes, the best frequency is the one that requires the fewest number of relay nodes on the backbone network. This will maximize performance of a network that must operate on a single frequency[ES.3]. The field "Number of Unconnected Links (NUL)" is the third most important consideration in selecting the best operating frequency. NUL is N*(N-1)/2 minus the known number of bidirectional links in the network. Thus, NUL represents the number of links that the

given topology falls short of being a fully connected network. Finally, the least significant component in the frequency selection metric is the index of the frequency in the sequential collection of candidate frequencies F_{sc}.



Description of Field	bits
a. n_{unc}	5
b. Number of relay nodes on backbone network, N_{bb}	5
c. NUL (Number of Unconnected Links).	10
d. Index of frequency in collection F_{sc}	6

Figure ES-1 Frequency selection Metric B.

Sample Outputs from the Network Frequency Selection Service

Figures ES-2 through ES-7 show sample outputs from the Network Frequency Selection Service for a 7-node simulation model with one mobile node (node 7), which traversed a pre-specified, deterministic path through the remaining nodes. There were a total of 12 frequencies tested; they are shown in table ES-1. Frequencies were tested in groups (NFSS test cycles) of four frequencies. The first three frequencies in each NFSS test cycle were taken (in round-robin fashion) from the table of candidate frequencies. The fourth frequency was taken as the best frequency from the previous test cycle. A "frequency test cycle" corresponds to a test of a single frequency. For the simulation tests, the NFSS test cycles were run contiguously. Normally, however, the NFSS test cycles would be separated by periods when the client network would be operational.

Each figure displays two NFSS test cycles, and each panel in figures ES-2 through ES-7 displays results for one frequency test cycle. The time-ordering of results is from top to bottom of the first column followed by the same ordering in the second column. Each panel shows the following information: an NFSS frequency test cycle identifier, the bidirectional links for that frequency test cycle, the value of the metric (in hexadecimal notation), the test frequency (in MHz), and the time at which the frequency test cycle began (in simulated hrs:min:sec.ms). For this example, the NFSS test cycle is approximately 100 seconds duration.

For the simulation model, frequency 12.03 MHz has the greatest communication range (115 km). The results show that this frequency is eventually chosen as the best frequency. The results also illustrate how the values of the metric change as platform 7 moves. To see what these metrics mean, we decode and compare the metric values 0x30282 and 0x230401, which appear in panels marked Nfss Tests 9.3 and 9.2, respectively, of figure ES-6. We write 0x30282 (after padding on the left to extend the result to the 26 bits required by the metric) = 00 0000 0011 0000 0010 1000 0010 (binary), which we group into the fields of our metric as follows: n_{unc} = 00000, N_{bb} = 00011, NUL = 0000001010, F_{sc} = 000010. In terms of deci-

mal integers, the metric value 0x30282 represents the situation where all nodes are connected (n_{unc} = 0), three relay nodes (nodes 3, 5, and 6) are required (N_{bb} = 3), the network is 10 links shy of being fully connected (NUL = 10), and the frequency index is 2 (F_{sc} = 2), which means that the test frequency is 12030 kHz. For the metric value 0x230401, we get 0x230401 = 00 0010 0011 0000 0100 0000 0001. This decodes to the following: n_{unc} = 00001, N_{bb} = 00011, NUL = 0000010000, F_{sc} = 000001. Thus, in terms of decimal integers, the metric value 0x230401 represents the situation where one node (node 5) is disconnected from the main body (n_{unc} = 1), three relay nodes (nodes 3, 5, and 6) are required (N_{bb} = 3), the network is 16 links shy of being fully connected (NUL = 16), and the frequency index is 1 (F_{sc} = 1), which means that the test frequency is 11062.5 kHz.

Table ES-1: Frequencies, channel noise levels, and communication ranges for test 7NFSE-T3.

Freq. Array Index	Freq.(Khz)	Channel Noise(db)	Range[a] (Km)
0	10680.0	-100.0	37.97
1	11062.5	-115.0	81.72
2	12030.0	-135.0	115.38
3	12673.5	-125.0	109.59
4	13238.5	-110.0	50.0
5	13980.0	-125.0	98.6
6	14695.0	-105.0	36.6
7	16923.0	-135.0	77.35
8	16954.0	-120.0	60.35
9	17514.0	-125.0	73.57
10	18556.0	-130.0	67.22
11	20025.0	-110.0	29.75

a. Range is based on the noise at the receiver, which is approximated as the larger of the channel noise (see column 3) and the receiver noise (-125 db).

Conclusions

This report describes a prototype Network Frequency Selection Service. The NFSS can be used as a stand alone system or it can be used to find the best operating frequency for a client network. As a stand alone system the NFSS can be used to monitor and evaluate the suitability of various HF channels for use in intratask force communication networks. Results of these tests could be analyzed manually to select the operational frequency for an intratask force network. Alternatively, the NFSS can be integrated with a client network so that the NFSS periodically tests a set of frequencies and automatically sets the transmitter/receiver to the best frequency for that network. Extensions of this work to cover the selection of multiple frequencies in multi-channel networks, such as those described in [ES.3] to [ES.5] are suggested.

References

[ES.1] Olsen, D. E., "MINCAP/HAMA Battle Group Network Protocol Description for the Shared Adaptive Internetworking Technology (SAINT) Program," NOSC Technical Report 1417, Naval Ocean Systems Center, San Diego, CA 92152-5000, March 1991.

[ES.2] "Military Standard Interoperability and Performance Standards for Medium and High Frequency Radio Equipment," MIL-STD-188-141A, 15 September 1988.

[ES.3] Thoet, William A., Dennis J. Baker, and Dennis N. McGregor, "A Multi-Channel Architecture for Naval Task Force Communication," NRL Memo Report TBD, Naval Research Laboratory, Washington, D. C. 20375, 1993.

[ES.4] Baker,.Dennis J., James P. Hauser, Dennis N. McGregor, and James T. Ramsey, "The UNT/NRL HF Intratask Force Communication Network Experiment," NRL Memo Report 6965, Naval Research Laboratory, Washington D. C. 20375, 1992.

[ES.5] Baker, Dennis J., James P. Hauser, and Dennis McGregor, "Design and Performance of an HF Multichannel Intratask Force Communication Network," NRL Report 9322, Naval Research Laboratory, Washington, D. C. 20375, Nov 12, 1991.

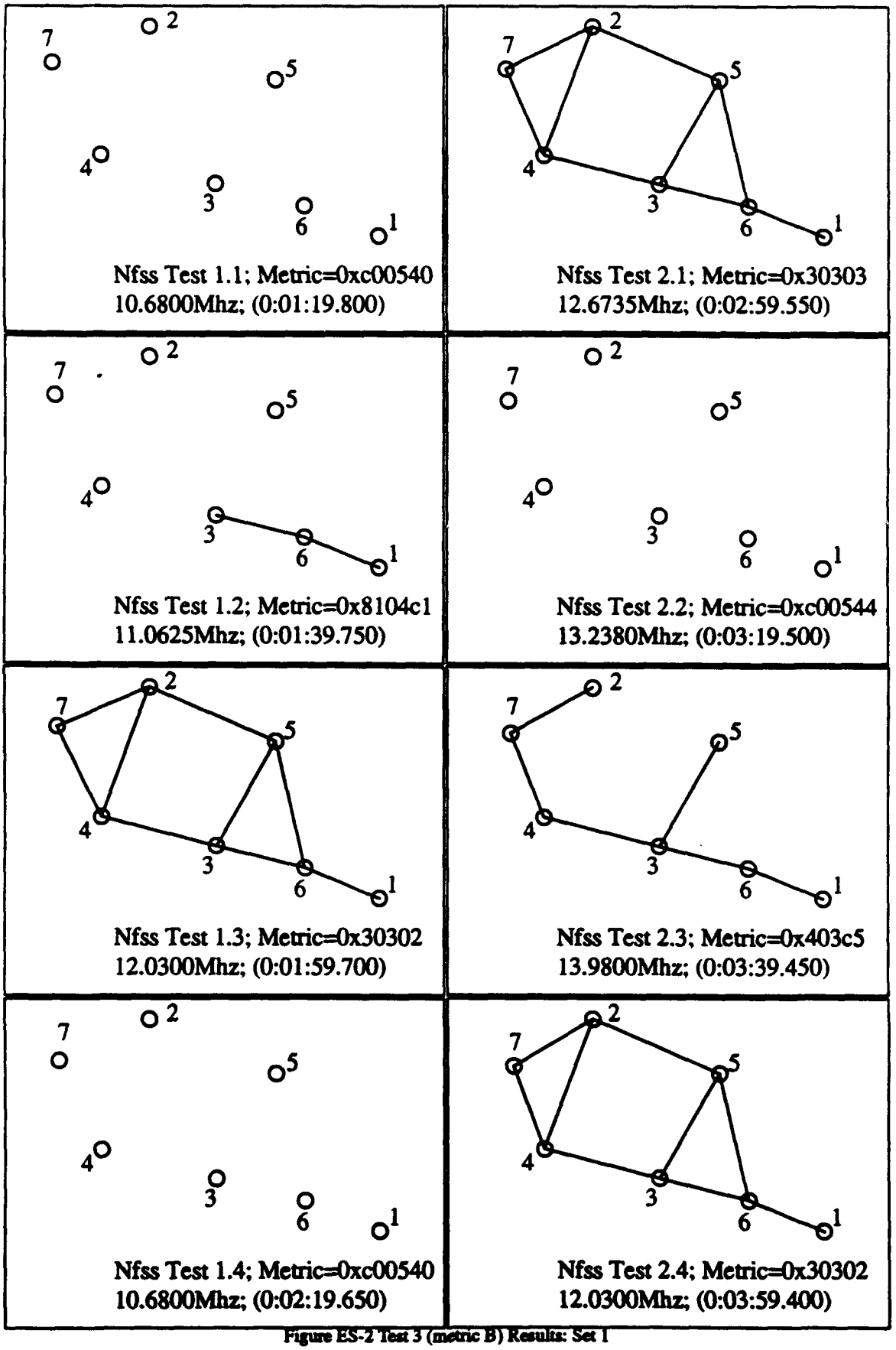
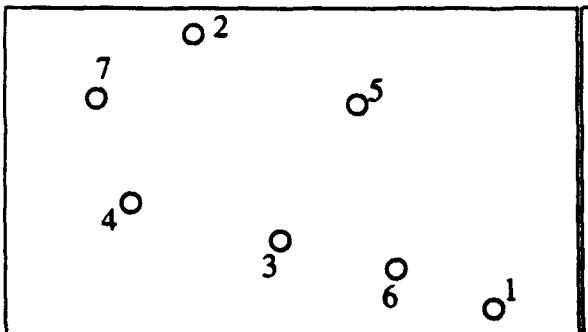
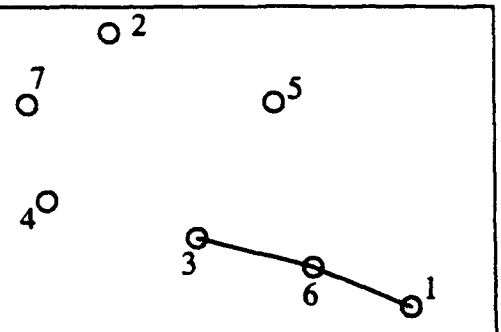


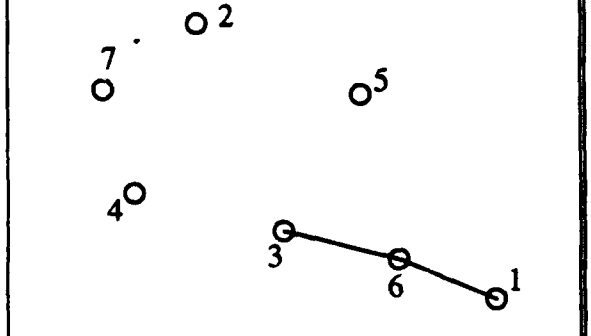
Figure ES-2 Test 3 (metric B) Results: Set 1



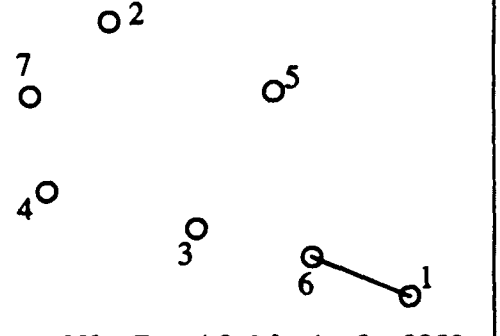
Nfss Test 3.1; Metric=0xc00546
14.6950Mhz; (0:04:39.300)



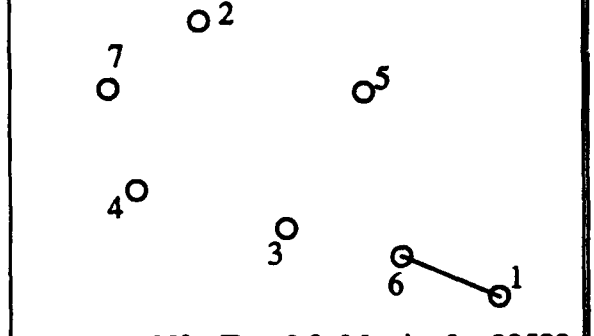
Nfss Test 4.1; Metric=0x8104c9
17.5140Mhz; (0:06:19.050)



Nfss Test 3.2; Metric=0x8104c7
16.9230Mhz; (0:04:59.250)

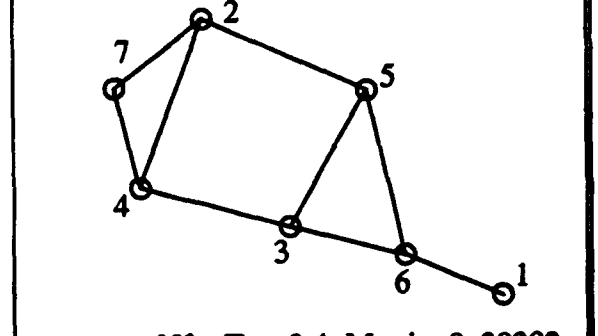


Nfss Test 4.2; Metric=0xa0050a
18.5560Mhz; (0:06:39.000)

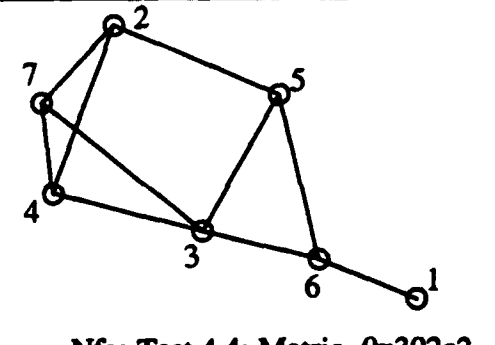


Nfss Test 3.3; Metric=0xa00508
16.9540Mhz; (0:05:19.200)

Nfss Test 4.3; Metric=0xc0054b
20.0250Mhz; (0:06:58.950)



Nfss Test 3.4; Metric=0x30302
12.0300Mhz; (0:05:39.150)



Nfss Test 4.4; Metric=0x302c2
12.0300Mhz; (0:07:18.900)

Figure ES-3 Test 3 (metric B) Results: Set 2

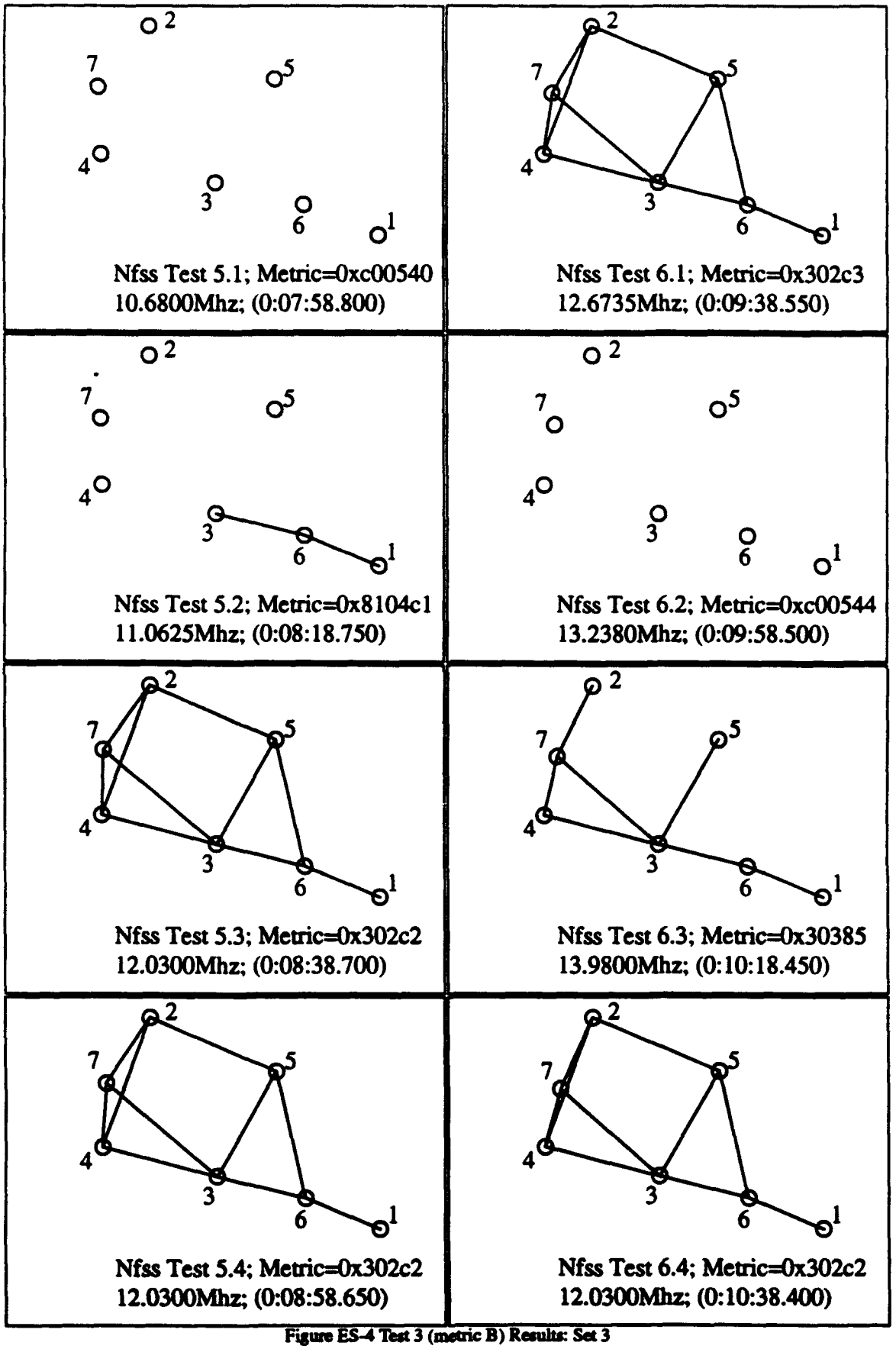


Figure ES-4 Test 3 (metric B) Results: Set 3

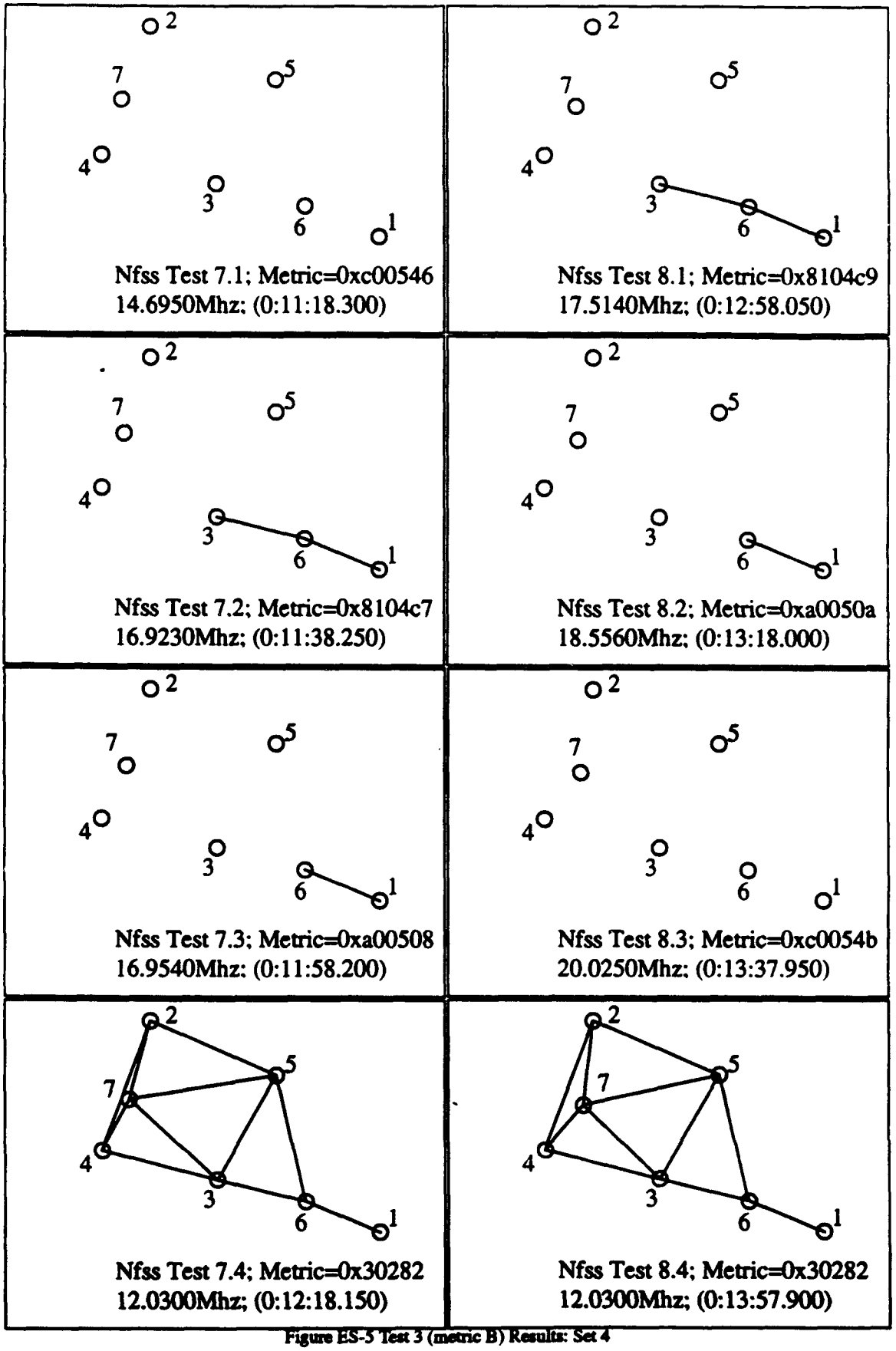


Figure ES-5 Test 3 (metric B) Results: Set 4

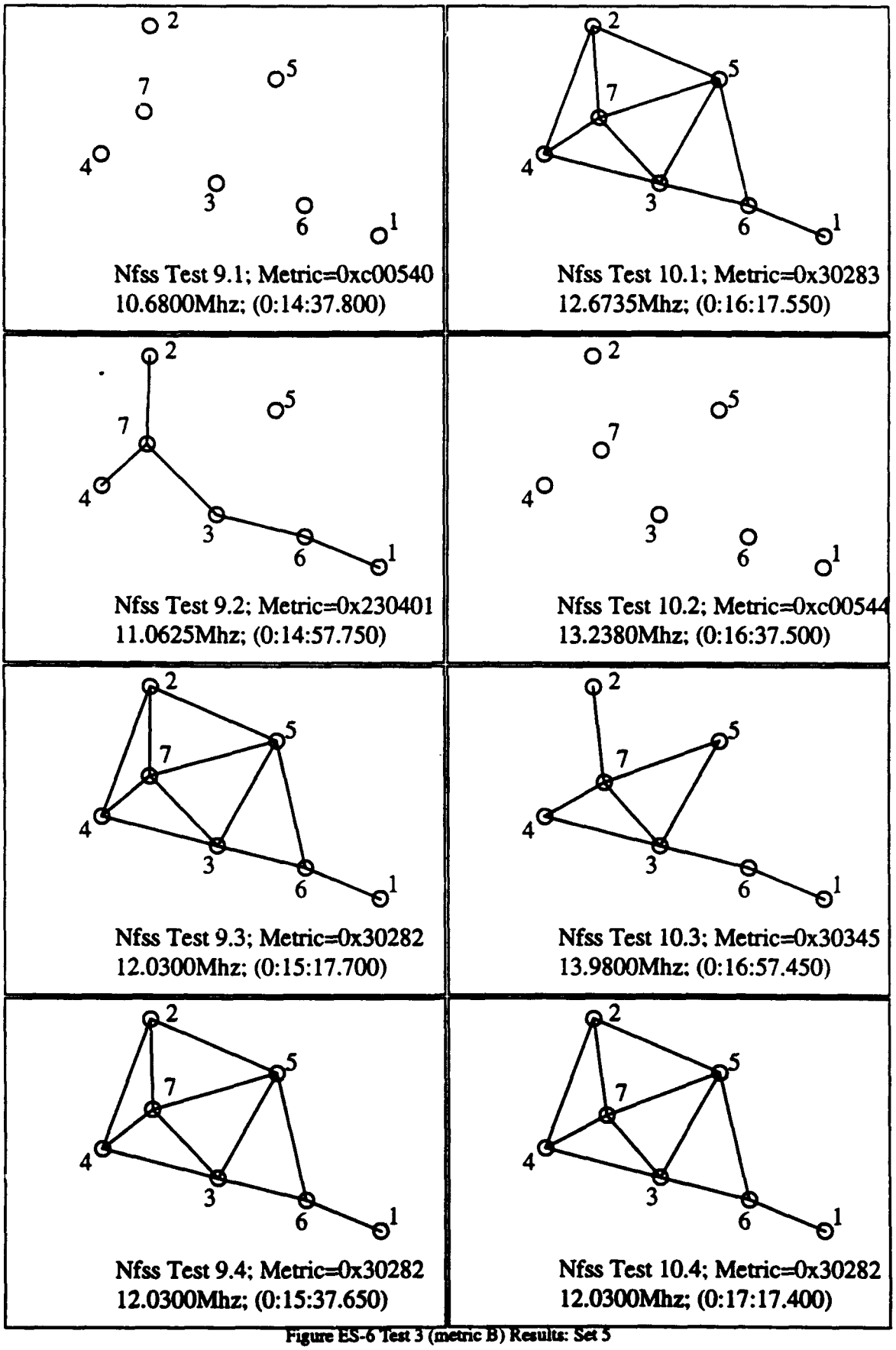


Figure ES-6 Test 3 (metric B) Results: Set 5

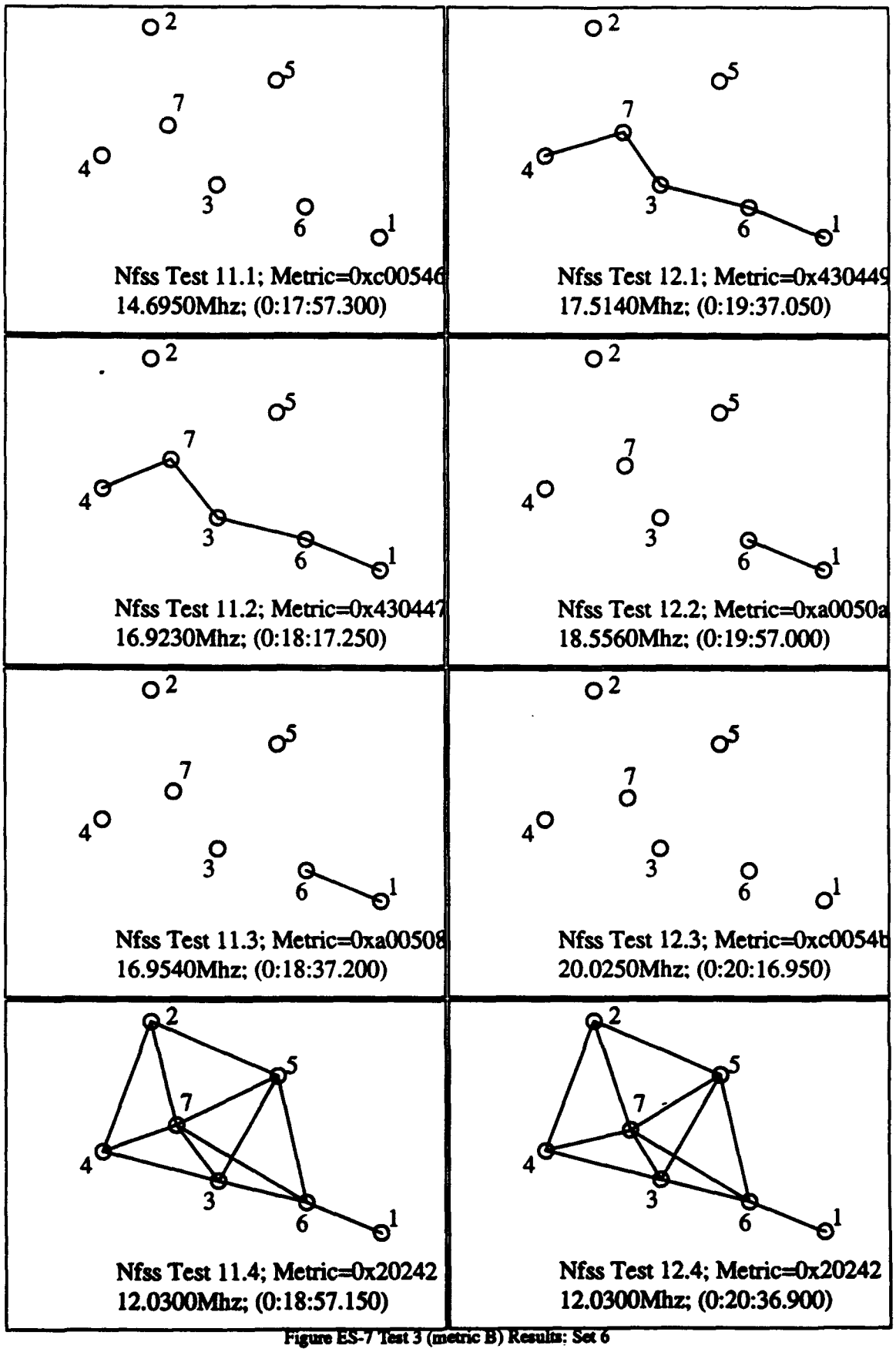


Figure ES-7 Test 3 (metric B) Results: Set 6

DESIGN, DEVELOPMENT, AND TESTING OF A NETWORK FREQUENCY SELECTION SERVICE (NFSS)

1. INTRODUCTION

The Naval Research Laboratory (NRL) has proposed the development of a Network Frequency Selection Experiment (NFSE) as part of the Naval Command Control and Ocean Surveillance Center - Naval Research and Development's (NCCOSC-NRaD) 6.2 Communications and Networking block. The objective of the NFSE is to demonstrate the capability of a High Frequency (HF) Intra Battle Group (IBG) communication network to automatically adapt its operating frequency to the best choice from a pool of candidate operating frequencies. HF (2 to 30 MHz) communications are subject to many time-varying effects such as Battle Group platform motion, day/night channel effects, changing sea state conditions, frequency versus communication range capability, jamming, and dynamic noise environments. These conditions all contribute to making it difficult to find a single, "best" frequency for network operation. Standardized Automatic Link Establishment (ALE) techniques [2] provide good frequency selection for a single link but do not answer the question of what is the optimum frequency choice for an entire network.

NRL has proposed to implement an automated Network Frequency Selection Service (NFSS) by enhancing the network control algorithms developed and successfully tested in the Unified Networking Technology (UNT) project [3][4]. This technology shall be demonstrated using the various nodes at the NCCOSC-NRaD UNT testbed. Since NRL's networking algorithms were successfully demonstrated during the September 1990 UNT tests, our modifications of that design to implement a frequency selection service should not be a major risk item. Once the techniques for automated selection of a netwide operating frequency have been successfully demonstrated, the algorithms can be incorporated into other systems, such as NCCOSC-NRaD's own MINCAP system.

2. ORGANIZATION OF REPORT

This report contains the documentation needed to guide the development of all software for the Network Frequency Selection Experiment, which, hereafter, is called the "NFSE System". This documentation loosely follows the software documentation requirements mandated by DOD-STD-2167A for software development[6]. Section 3 lists applicable documents. Section 4 defines acronyms, symbols, special terms, and software constants used in this report. Sections 5 through 11 are organized so that each major section covers a key Data Item Description (DID) of the 2167A standard. Thus, section 5 corresponds to the System Specification for the NFSE system. Section 6 corresponds to the System Design Document. Section 7 contains the Software Development Plan. Section 8 corresponds to a Software Requirements Document for each Computer Software Configuration Item (CSCI). Section 9 corresponds to an Interface Requirements Specification for interconnecting CSCIs. Section 10 provides the equivalent of a Software Design Document for each CSCI; and section 11 corresponds to an Interface Design Document for each CSCI. Sections 12, 13, and 14 cover the in-lab test plans, descriptions, and results, respectively.

The information in this report is subject to change. To allow for these changes, Sections 4 through 11 each end in a subsection that is intended to contain the revision history for that section. The initial version of this report is Version 1.0. The current version number shall always appear under the title on the report's title page.

Manuscript approved December 10, 1993.

3. APPLICABLE DOCUMENTS

3.1 Government Documents

[1] "Navy Exploratory Development Program FY 1993 Technology Block Plan: Communications and Networking (CS2B)," NCCOSC-NRaD, 9, September 1992.

[2] "Military Standard Interoperability and Performance Standards for Medium and High Frequency Radio Equipment," MIL-STD-188-141A, 15 September 1988.

[3] Baker, Dennis J., James P. Hauser, and Dennis McGregor, "Design and Performance of an HF Multichannel Intratask Force Communication Network," NRL Report 9322, Naval Research Laboratory, Washington, D.C. 20375, Nov 12, 1991.

[4] Baker, Dennis J., James P. Hauser, Dennis N. McGregor, and James T. Ramsey, "The UNT/NRL HF Intratask Force Communication Network Experiment," NRL Memo Report 6965, Naval Research Laboratory, Washington D.C. 20375, 1992.

[5] Olsen, D. E., "MINCAP/HAMA Battle Group Network Protocol Description for the Shared Adaptive Internetworking Technology (SAINT) Program," NOSC Technical Report 1417, Naval Ocean Systems Center, San Diego, CA 92152-5000, March 1991.

[6] DOD-STD-2167A, Military Standard: Defense System Software Development, February 29, 1988.

[7] Lindquist, D., "UNT NRL HF Link Controller Project Design Work Book 3-Node Demonstration," Preliminary Draft, September 1989.

[8] "Software Requirements Specification (SRS) for the Naval Research Laboratory High Frequency Link Controller," Initial Draft, September 1989.

[9] "Program Listings Manual (PLM) for the Naval Research Laboratory UNT High Frequency Link Controller," October, 1990.

[10]"Project Design Manual (PDM) for the Naval Research Laboratory UNT High Frequency Link Controller," December, 1990.

[11]Hauser, James P. and Dennis J. Baker, "System Design and Software Specification for the High Frequency Intra Battle Group Communication System Network Controller for the Unified Networking Technology Demonstration," 8 September 1989.

3.2 Non-Government Documents

[12]"Creating a VxWorks Network Interface Driver," Technical Note, Wind River Systems, Inc., August 1991.

[13]Jarman, G. B., "Timing Characteristics for the UNT Experiment's HF Modem," Harris Corp., Govt. Comm. Sys. Div., Encl of ltr to J. Hauser (NRL), (3/7/89).

[14]User's Manual, Model 9390-5500, GPS Time/Frequency System, Datum Inc.

[15]Ascii Remote Control Option for RF-590A and R-2368/URR Receivers, Harris Corp., RF Communications Group, Publ. 10215-0034, May 1989.

[16]Internet Protocol, in DDN PROTOCOL HANDBOOK, Vol. Two, Defense Communications Agency, Dec. 1985.

4. DEFINITIONS

4.1 Acronyms and Abbreviations

Table 1: Meanings of acronyms and abbreviations

Acronym/ Abbreviation	Meaning
ALE	Automatic Link Establishment
ASAP	As Soon As Possible
BLC	Black Link Controller
COTS	Commercial, off-the-shelf
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSS	Communications Support System
CSU	Computer Software Unit
DID	Data Item Description
DOD-STD	Department of Defense Standard
DRAM	Dynamic Random Access Memory
DSPT	Distributed Simulation and Prototyping Testbed
EBB	Enhanced Backbone Network
GMT	Greenwich Mean Time
GPS	Global Positioning System
GUI	Graphical User Interface
HAMA	Handoff Assigned Multiple Access
HF	High Frequency
HW	Hardware
IBG	Intrabattle Group

Table 1: Meanings of acronyms and abbreviations

Acronym/ Abbreviation	Meaning
IP	Internetwork Protocol
IRS	Interface Requirements Specification
ISO	International Organization for Standardization
KG	Key Generator
LCA	Linked Cluster Architecture or Linked Cluster Algorithm
LSB	Least Significant Bit
MHz	Megahertz
MIL-STD	Military Standard
MINCAP	Minimum Coverage Approximation
MSB	Most Significant Bit
mo	month
ms	milliseconds
NCCOSC-NRaD	Naval Command Control and Ocean Surveillance Center - Naval Research and Development (formerly NOSC)
NFSE	Network Frequency Selection Experiment
NOSC	Naval Ocean Systems Center (now NCCOSC-NRaD)
NRL	Naval Research Laboratory
n.a.	not applicable
nm	nautical miles
ONT	Office of Naval Technology
OOD	Object oriented design
OOP	Object oriented programming
PDM	Project Design Manual
PLM	Project Listings Manual
ppm	Pulse per millisecond
Qty.	Quantity
RB	Reliable Bidirectional
RCS	Revision Control System

Table 1: Meanings of acronyms and abbreviations

Acronym/ Abbreviation	Meaning
RCVR	Receiver
RF	Radio Frequency
RLC	Red Link Controller
SAINT	Shared Adaptive Internetworking Technology
SDD	Software Design Document
SDF	Software Development Files
SDL	Software Development Library
SDP	Software Development Plan
SRS	Software Requirements Specification
SSDD	System/Segment Design Document
SSS	System/Segment Specification
STD	Software Test Descriptions
SW	Software
s	seconds
T/R	Transmit/Receive
TBD	To Be Determined
UNT	Unified Networking Technology
XMTR	Transmitter
yr	year

4.2 Identifiers Used to Describe System

Table 2: CSCI Identifiers

Identifier	Long Name
NC	Network Controller
LC	Link Controller
SC	System Controller

Table 3: CSC Identifiers

Identifiers	Long Name	Parent CSCI
LC_BIOS	Basic Input Output System	LC
LC_OS	LC Operating System	LC
LC_TC	LC Timing Control	LC
NC_CC	Congestion Control	NC
NC_IP_CONN	Internet Protocol Connection	NC
NC_LC_CONN	Link Controller Connection	NC
NC_MR	Message Router	NC
NC_NCL	Network Connectivity Learning	NC
NC_NFSS	Network Frequency Selection Service	NC
NC_NM	Network Manager	NC
NC_NRT_SU	Startup for NC_NRT Processor	NC
NC_NS	Network Structuring	NC
NC_RBM	Receive-Buffers Management	NC
NC_RC	Receiver Control	NC
NC_RTO_CL	Real-time Output Client	NC
NC_RTO_SV	Real-time Output Server	NC
NC_SRC_SNK	Traffic Source/Sink (for simulated user-traffic)	NC
NC_TBM	Transmit-Buffers Management	NC
NC_TC	NC Timing Control	NC
NC_XC	Transmitter Control	NC

Table 4: CSU Identifiers

Identifier	Long Name	Parent CSC
BTRA	Broadcast Traffic Routing Algorithm	NC_MR
CCA	Congestion Control Algorithm	NC_CC
LCA	Linked Cluster Algorithm	NC_NS

Table 4: CSU Identifiers

Identifier	Long Name	Parent CSC
NCLA	Network Connectivity Learning Algorithm	NC_NCL
NFSA	Network Frequency Selection Algorithm	NC_NFSS
NSMA	Network Status Monitoring Algorithm	NC_NM
P2PRA	Point-to-Point Routing Algorithm	NC_MR
PPA	Periodic Probing Algorithm	NC_NCL
RBMA	Receive-Buffers Management Algorithm	NC_RBM
RSA	Receiver Scheduling Algorithm	NC_RC
TBMA	Transmit Buffers Management Algorithm	NC_TBM
TSA	Transmission Scheduling Algorithm	NC_XC

Table 5: Interface Identifiers

Identifier	Long Name	Components Linked	
IF_LC_GPS	LC/GPS Interface	LC	GPS
IF_LC_LC	LC Peer-Level Interface	LC	LC
IF_LC_NOSC	LC/NOSC BLC Interface	LC	NOSC BLC
IF_LC_RCTL	LC/Receiver Control Interface	LC	Receiver
IF_LC_XCTL	LC/Transmitter Control Interface	LC	Transmitter
IF_NC_LC	NC/LC Interface	NC	LC
IF_NC_NC	NC Peer-Level Interface	NC	NC
IF_OP_SC	Operator/SC Interface	Operator	SC
IF_SC_LC	SC/LC Interface	SC	LC
IF_SC_NC	SC/NC Interface	SC	NC

Table 6: VME-Board Identifiers

Identifier	Long Name
LANCNTR	Ethernet Controller Board (ENP-10L)
M224NRL	NRL's MVME224A-1 Memory Board

Table 6: VME-Board Identifiers

Identifier	Long Name
M333NRL	NRL's MVME333-2 Processor Board
M333NRaD	NRaD's MVME333-2 Processor Board
NC_NRT	NC Non-Real-time Processor Board (MVME135A)
NC_RT	NC Real-time Processor Board (MVME135A)
Proc #n	Processor #n (n is the vxWorks-assigned #)

Table 7: Names of System Capabilities

Identifier	Long Name
IN_CAP	Internal Network Capability
MCA_CAP	System Monitor, Controller, and Archiver
NC_COM	
ND_TIME	Node Timing
NFSS	Network Frequency Selection Service
SC_ARCHIVE	System Archiver
SC_MONITOR	System Monitor
TPD_CAP	Test-Plan Description Service

4.3 Symbols

Table 8: Symbols used in this document (except for time-of-occurrence symbols).

Notation	Page	Meaning	Min.	Max.	Units
%	87	An operator such that i%j returns the remainder of the division of i by j.	n.a.	n.a.	none
A_i	26	*i*th scheduled active cycle of NFSS	n.a.	n.a.	n.a.
at (*j*)	87	Function that returns pointer to the *j*th element of a sequential collection.	n.a.	n.a.	n.a.
F_{sc}	18	Sequential collection of candidate frequencies given as input to NFSS. $F_{sc} \equiv \{f_j\}_{sc}$	f_{min}	f_{max}	MHz

Table 8: Symbols used in this document (except for time-of-occurrence symbols).

Notation	Page	Meaning	Min.	Max.	Units
$fa[]$	25	Array of test-frequencies to be used by NFSS during current activation.	f_{min}	f_{max}	MHz
f_b	18	Frequency chosen by the NFSS as the "best" frequency.	f_{min}	f_{max}	MHz
$f_{b,i}$	87	Frequency chosen as the "best" frequency during the ith active cycle of the NFSS.	f_{min}	f_{max}	MHz
f_{max}	19	Maximum value for frequencies in the collection F_{sc}	unspec.	unspecified	MHz
f_{min}	19	Minimum value for frequencies in the collection F_{sc}	unspec.	unspecified	MHz
$\{f_j\}_{sc}$	18	Sequential collection of candidate frequencies given as input to NFSS.	f_{min}	f_{max}	MHz
$LQI_{i,j}$	65	(Link Quality Index) - measure of the quality of link from node j to node i	0	$2^{NB\_SINGLE\_LQI} - 1$	none
LQI_r	66	Threshold value for classifying a link as unreliable or reliable	0	$2^{NB\_SINGLE\_LQI} - 1$	none
M	25	Number of elements in array $fa[]$	1	FA_MAX	none
N	18	Number of nodes in client network	2	N_{max}	none
N_F		Number of frames in all but the last NFSS frequency test cycle P_j	4	unspecified	none
N_{max}	19	Maximum value of N	7	31	none
P_j	19	NFSS frequency test cycle j (using frequency stored in $fa[j]$	n.a.	n.a.	n.a.
$sizeof(F_{sc})$	87	Number of elements in the collection F_{sc}	2	unspecified	none
T		Denotes a duration	0	unspecified	ms
T_A		The duration over which the NFSS is active.	TBD	TBD	ms
T_{AP}	18	Period between start of successive, scheduled active cycles of the NFSS	TBD	TBD	ms
T_{P_j}	26	Duration over which the NFSS is testing with the frequency in $fa[j]$	TBD	TBD	ms
T_{RP}	59	Reorganization period for internal network	TBD	unspecified	ms

Table 8: Symbols used in this document (except for time-of-occurrence symbols).

Notation	Page	Meaning	Min.	Max.	Units
T_{fill}		See figure 19.	0	TBD	ms
T_{sud}	16	Minimum delay from NFSE System startup until $t_{a,s}$.	0	TBD	ms
t	15	Denotes a time-of-occurrence, referenced to some point in time.	0	TBD	ms
$\bar{t}$	15	Denotes a time-of-occurrence, referenced to calendar time.	unspec	unspecified	day:mo:yr:ms

Table 9: Time-of-occurrence symbols used in this document.

Notation	Page	Meaning	Ref. to	Units
$\bar{t}_0$	15	Calendar-time reference for the system	calendar	day:mo:yr
t_{P_j}	26	Start of frequency test cycle P_j	$t_{a,l}$	ms
$t_{a,i}$	17	ith scheduled active cycle for the NFSS.	$\bar{t}_0$	ms
$t_{a,l}$		Start of last (most recent) active cycle of NFSS	$\bar{t}_0$	ms
$t_{a,s}$	16	Start of first active cycle of NFSS	$\bar{t}_0$	ms
t_{fm}	26	Time of computation of frequency selection metric	t_{P_j}	ms
t_{su}	16	System-referenced-time that system startup occurs	$\bar{t}_0$	ms
$\bar{t}_{su}$	15	Calendar-time that system startup occurs	calendar	day:mo:yr:ms

Table 10: Software constants.

Name of Constant	Value
CCA_EPOCHS_BETWEEN_INITS	2
CCA_MAX_NUM_CL_SCHEDS_REPORTED	5
CCA_INIT_GLOBAL_FACTOR	1.0
F_MAX	30.0 MHz

Table 10: Software constants.

Name of Constant	Value
F_MIN	2.0 MHz
LQI_RELIABLE	3
MAX_NET_SIZE	31 nodes
MAX_NUM_RCVRS	5 per node
MAX_NUM_SCHEDS_REPORTED	3
MAX_SEQ	511
MAX_USER_DATA	792 bits
MSG_STORE_SZ	64 msgs
M_BLKSIZE	22 bytes
NB_GLB_CONGESTION_FACTOR	8 bits
NB_NCLA_EPOCH	4 bits
NB_NODE_ID	5 bits
NB_NUM_SCHEDS_REPORTED	2 bits
NB_SINGLE_LQI	2 bits
NB_SEQ	9 bits
NB_SLOTS_REQUESTED	8 bits
NB_STATUS	2 bits
NB_TRAFFIC_LIMIT_REQUESTED	5 bits
NCLA_MAX_EPOCH	4
NCLA_MAX_EPOCHS_BETWEEN_RETRANSMISSIONS	20
NUM_FRAMES_PER_EPOCH	4
RAMP	0.1
SWITCH	1.04

4.4 Terms

Table 11: Terms defined in this document.

Term	Meaning
Calendar time	GMT, as might be obtained with a GPS receiver
Client network	Network that uses the Network Frequency Selection Service

Table 11: Terms defined in this document.

Term	Meaning
Host node	The node at which software is executing.
Mode A	System mode characterized by interleaved operation of NFSS and external client network.
Mode B	System mode characterized by interleaved operation of NFSS and internal client network.
Mode C	System mode characterized by concurrent operation of NFSS and internal client network.
System reference time	NFSE System reference time (0 hrs: 0 min: 0 secs Jan. 1, 1970 GMT in our implementation)

4.5 Revisions to Section 4

None.

5. SYSTEM SPECIFICATION

This section serves as the System/Segment Specification (SSS) for the Network Frequency Selection Experiment (NFSE) System.

5.1 Scope

5.1.1 Identification

This system shall be referred to as the Network Frequency Selection Experiment System or as the NFSE System.

5.1.2 System Overview

This system shall implement a field-demonstration of a Network Frequency Selection Service NFSS). This service shall periodically test network connectivities and set the client network's transmitter and receiver to the "best" frequency for network operation. The latter shall be chosen from among a collection of allowed frequencies, which is provided as input to the NFSS.

5.1.3 Section Overview

Section 5 specifies the requirements for the NFSE System and forms the Functional Baseline for that system.

5.2 Applicable Documents

See Section 3.

5.3 System Requirements

5.3.1 Definitions

5.3.1.1 General Description

To understand the NFSE System requirements and constraints, it is first necessary to provide some background into how the UNT tests were performed.

Two HF network communication architectures were field-tested during the UNT tests. NRL tested a multi-frequency network architecture called the *Linked Cluster Architecture* (LCA)[3] and [4] and NOSC tested a single-frequency network architecture based on the HAMA/MINCAP protocols [5]. Both the NRL and NOSC HF networks used the same set of modems, transmitters, receivers, power amplifiers, and antennas. In addition, both network controllers were co-resident in the same VME-chassis, although they ran on separate VME boards within that chassis. One major difference in the hardware configurations for the two networks is that the NOSC HF network also incorporated a link encryption device. Hence the NOSC hardware architecture also included a KG84 crypto, which was followed by another VME chassis in which they implemented their "Black Link Controller" (BLC). In order for both NRL and NOSC to perform their respective tests, it was necessary to alternate the days that each organization could use the UNT

equipment suite. In addition, a small amount of cabling changes had to be made each time a different group needed to use the UNT equipment. What is important, from the standpoint of developing a network frequency selection experiment, is that in the UNT project, NOSC implemented a single-channel network, which could have benefitted from the availability of a network frequency selection service. On the other hand, NRL implemented a network that included many of the components needed to construct and test such a service.

5.3.1.2 Missions

The purpose of the NFSE System is to demonstrate a technique for selecting a single, "best" operating frequency for an HF radio network.

5.3.1.3 Operational Concepts

The NFSE System shall time-share the RF transmission assets with a *client network* for which it is providing the Network Frequency Selection Service. At predetermined time intervals the NFSE System will control the RF assets of the client network. During these periods, the NFSE System shall select the "best" operating frequency for the client network. Before returning control of the RF assets back to the client network, the NFSE System shall set the transmitter and receiver to operate at the selected frequency.

5.3.1.4 System Design Constraints/Considerations

The NFE System shall build on the hardware and software resources and testing experience obtained during the UNT tests. The advantages of this approach are the following: 1) NRL's UNT network already implements most of the components needed to support a Network Frequency Selection Service, 2) this approach re-uses large amounts of UNT hardware and software that has been field-tested, 3) it minimizes system integration costs, and 4) it takes advantage of software developed for archiving test results, duplicating test results in the laboratory, and performing additional (post-field-test) testing in the laboratory.

5.3.1.5 System Diagrams

Figure 1 shows the planned node hardware architecture in which the NFSE System is to be embedded. The major change from the UNT node architecture is to move the NRL Network Controller (NC) and Link Controller (LC) hardware and software from the RLC chassis to the BLC chassis and to connect the LC to the RF strings via the NOSC UNT hardware/software interfaces. It is also necessary to supplement the UNT design with an additional port for controlling the radio frequency of the transmitter. Also, an ethernet controller board must be added to the BLC so that a Sun workstation can be attached to the BLC. This workstation provides a storage medium for test results and also serves as a monitor and controller for the system.

Figure 4 is a context diagram for the NFSE System.

5.3.1.6 Some Notational Conventions

Here and throughout the remainder of this document, we frequently need to talk about the time-of-occurrence of various events. To facilitate these discussions we define what we mean by calendar-time and system-reference-time. *Calendar-time*, for the purposes of this paper, is Greenwich Mean Time (GMT) such as might be obtained from a GPS receiver. *System-reference-time* is a value of calendar-time that serves as

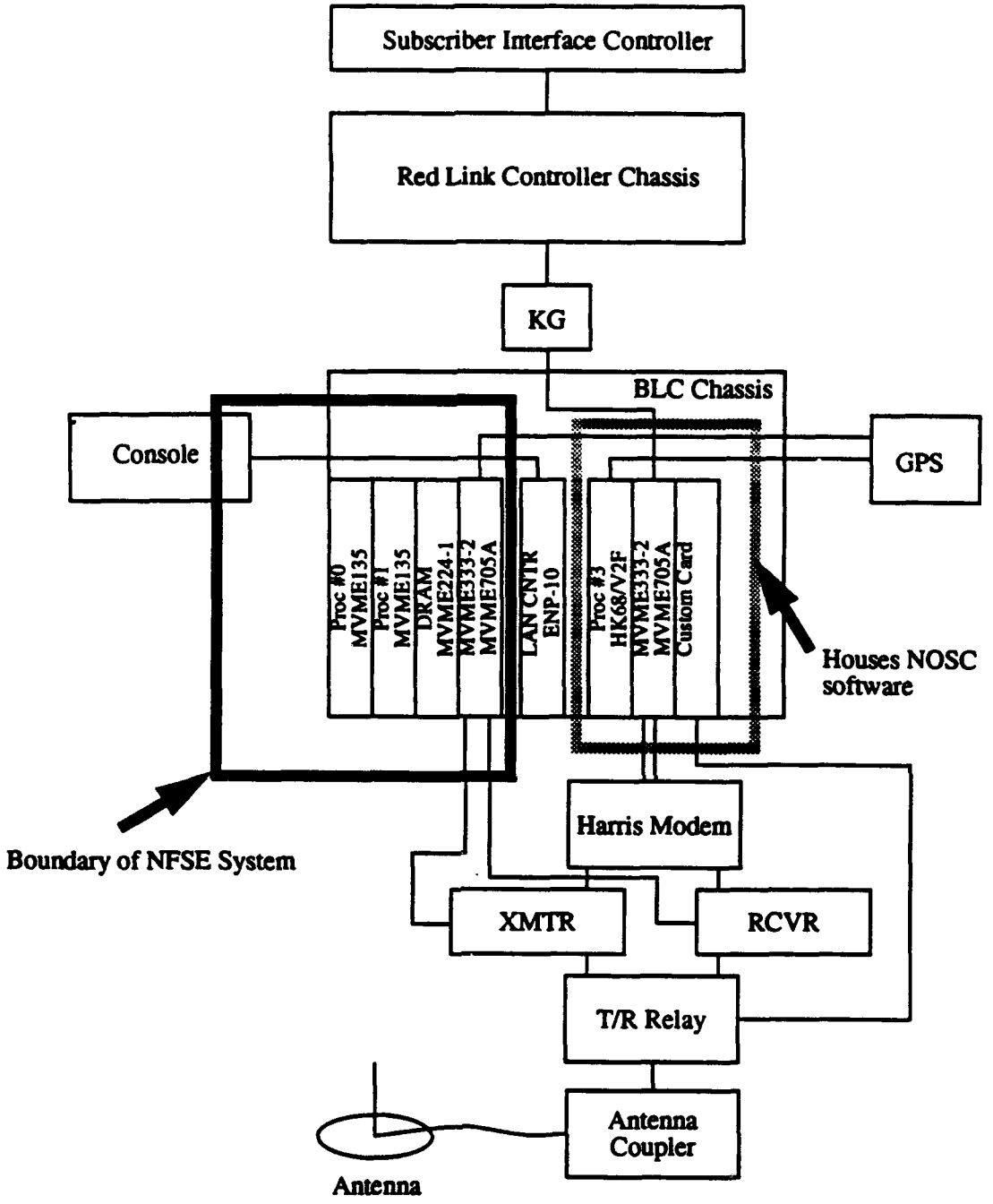


Figure 1 Node hardware architecture in which the NFSE System is to be embedded.

a basic time reference for the NFSE System; it is denoted by $\bar{t}_0$.

We use the convention of reserving lower case t to refer to the time-of-occurrence measured with respect to some reference time, which is either expressly stated or should be obvious from the context. When we wish to talk about time-of-occurrence in calendar-time, we shall use a "bar" over the t as in $\bar{t}$. Finally, when we are interested only in expressing a duration of time, we shall use an upper case T.

The system requires that the system startup time $\bar{t}_{su}$ must occur after the system reference time $\bar{t}_0$. Thus, the following must hold.

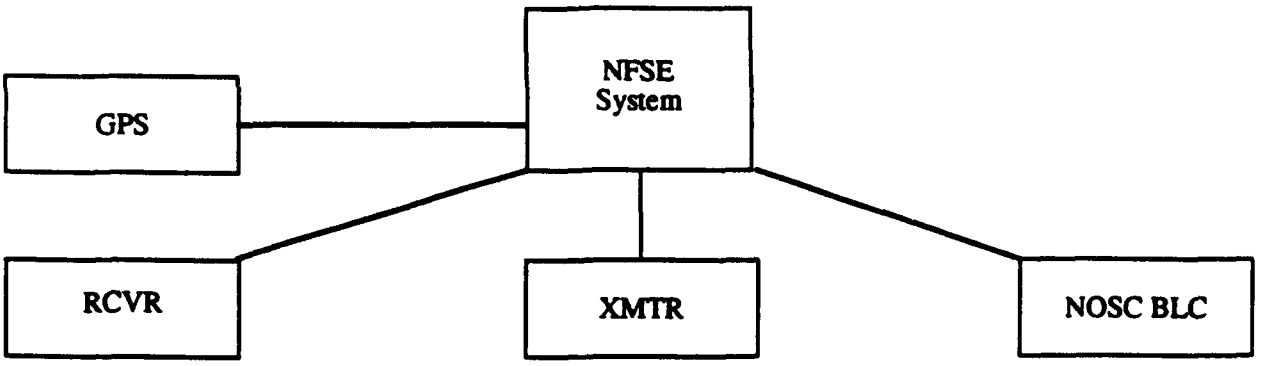


Figure 2 NFSE System context diagram.

$$\overline{t_0} < \overline{t_{su}} \qquad \text{(EQ 6)}$$

We also define N as the number of nodes in the client network.

5.3.2 Characteristics

When the NFSE System software is started at time t_{su}, it should compute the next available time $t_{a,s}$ for which the NFSS shall be active. $t_{a,s}$ shall fall on time boundaries as specified in Section 5.3.2.1.1.1.2. In addition, the implementation shall specify the time T_{sud}, which is the minimum allowed delay from system startup until $t_{a,s}$. Thus, the system imposes the following constraint.

$$t_{su} < t_{a,s} - T_{sud} \qquad \text{(EQ 7)}$$

The system shall have three operating modes, which define different ways in which the Network Frequency Selection Service can be exercised.These options are shown in figure 3. The mode of operation shall be determined at system initialization by reading a startup file. The mode of operation shall be changeable only by restarting the system with a different mode selected. These modes are defined as follows.

Mode A - In this mode of operation the NFSS and HAMA/MINCAP network time-share the RF communication assets. When the NFSS has control of the RF assets it performs those tasks needed to select the "best" operating frequency. At the end of its duration of control, the NFSS sets the transmitter to the new operating frequency. Then the HAMA/MINCAP network operates in its normal fashion using this operating frequency until it is time to repeat the cycle.

Mode B - This is identical to Mode A with the exception that the NFSE uses (optional) its own, internal network communication protocols instead of the HAMA/MINCAP protocols.This mode, without the optional activation of the internal network is used if one is solely interested in exercising the NFSS. The use or non-use of the internal network in this mode of operation shall be determined at system initialization by reading a startup file.

Mode C - This is identical to Mode B with the exception that, if the internal network is activated, network communication does not cease operation while the NFSS is operating.

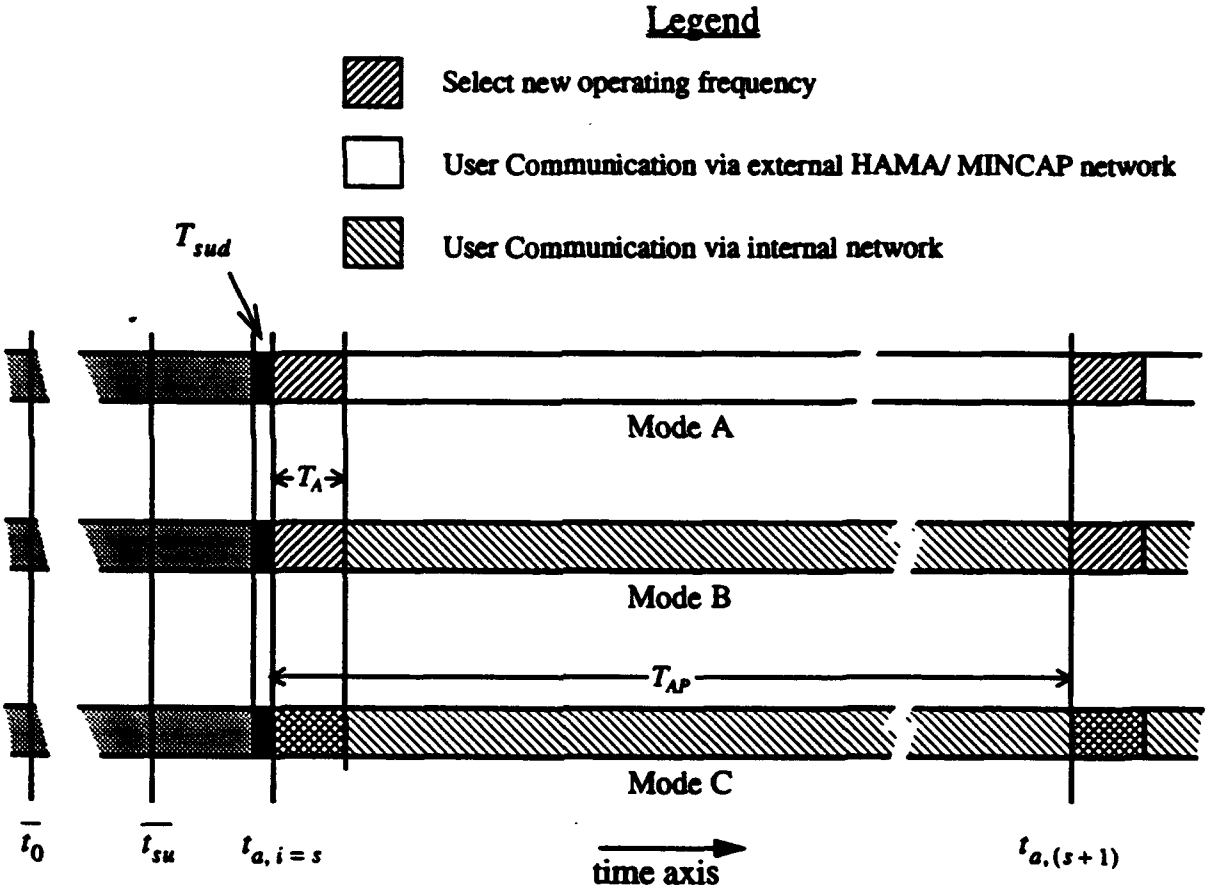


Figure 3 Modes of NFSE System operation: Mode A - Interleaved operation of NFSS and HAMA/MINCAP network; Mode B - Interleaved operation of NFSS and internal network: Mode C - Concurrent operation of NFSS and internal network.

5.3.2.1 Performance Characteristics

This subsection specifies the NFSE System's capabilities in the context of the modes in which the system can exist.

5.3.2.1.1 Mode A

Mode A interleaves the operation of the NFSS with that of an external network, such as a HAMA/ MINCAP network. [5]

5.3.2.1.1.1 Network Frequency Selection Service (NFSS)

5.3.2.1.1.1.1 Informal Description

The intended application of the NFSS capability is a single-frequency, HF radio network for an intratask force. An example of such a network is the HAMA/MINCAP network, which was demonstrated in the UNT tests. [5] For the intended application the communicating platforms are generally dispersed over an area less than 300 nm across. Communication in this scenario is usually achieved by HF groundwaves, although some skywave propagation is possible. Selection of an HF operating frequency for the network could be decided prior to starting up the network and made part of the communications plan. However, problems can arise if the pre-selected operating frequency results in poor communication performance, unless there is some method for switching to a more suitable frequency. The NFSS provides an automated procedure for selecting, from a given sequential collection[1] of HF frequencies, the "best" frequency for use by the network. After selecting the operating frequency, the NFSS proceeds to set the client network node's transmitter and receiver to the chosen frequency. The functions of the NFSS are implemented in software to perform this service. The NFSS is intended to be used as a scheduled service. That is, it begins and ends operation at predetermined times. (See note 1 in Section 5.6).

5.3.2.1.1.1.2 Formal Description

1. The NFSS can only control the client network's transmitter and receiver at specific times (referenced to t_0) that satisfy

$$t_{a,i} = iT_{AP} \qquad \text{(EQ 8)}$$

   where $i = 0,1,2,...$ and T_{AP} is given.

2. When activated at time $t_{a,i}$, the NFSS shall attempt to select a common, "best" frequency f_b from a given, non-empty sequential collection $F_{sc} \equiv \{f_j\}_{sc}$.
3. At the time $(t_{a,i} + T_A)$ the NFSS shall cease operation, leaving the client's transmitter and receiver set to operate on f_b.
4. The minimum period between the start of successive activations of the NFSS, T_{AP}, must satisfy

$$T_{AP} \geq T_A. \qquad \text{(EQ 9)}$$

5.3.2.1.1.1.3 Assumptions

1. An implementation shall specify the minimum allowed value for T_A
2. An implementation shall specify the maximum number of elements in the sequential collection $\{f_j\}_{sc}$.
3. An implementation may provide additional means for notifying a client of the "best" frequency f_b. For example, f_b may be loaded into a memory location that the client can read.

---

1. A *sequential collection* is defined as a collection whose members can be accessed via an index.

4. An implementation shall specify the minimum, f_{min}, and maximum, f_{max}, values for frequencies in the collection $\{f_j\}_{sc}$

5.3.2.1.1.2 Monitor, Controller, and Archiver Capability (MCA_CAP)

The Monitor, Controller, and Archiver capability is a composite of several capabilities. The NFSE System Archiver capability provides a log of the operation of the NFSE System. This log consists of several files that first are written to the console's disk and subsequently are transferred to a permanent storage medium, such as magnetic tape. The intent is that this log should be of sufficient detail to permit duplication of field-test results in the laboratory, as was done for the NRL UNT tests. [4] The NFSE System also provides a system monitoring and control capability, which is accessible to the system operator. The interface to this service shall provide the operator with a means of ascertaining the status of the NFSE System and for starting and stopping the system.

5.3.2.1.1.3 Test-Plan Description Capability (TPD_CAP)

This capability shall provide the means to specify the tests to be performed with the NFSE System.

5.3.2.1.2 Mode B

Mode B is like Mode A with the exception that the NFSE uses (optionally) its own, internal network to pass user traffic. It is intended that Mode B be used for initial tests of the NFSE System prior to final tests using Mode A. This mode can be used without the optional internal network, if one is interested solely in exercising the Network Frequency Selection Service.

5.3.2.1.2.1 Network Frequency Selection Service (NFSS)

See Section 5.3.2.1.1.1.

5.3.2.1.2.2 Monitor, Controller, and Archiver Capability (MCA_CAP)

See Section 5.3.2.1.1.2

5.3.2.1.2.3 Internal Network Capability (IN_CAP)

The internal network shall be capable of simulating user traffic or passing real traffic that it receives from other networks via the Internet Protocol (IP).

5.3.2.1.2.4 Test-Plan Description Capability (TPD_CAP)

See Section 5.3.2.1.1.3.

5.3.2.1.3 Mode C

5.3.2.1.3.1 Network Frequency Selection Service (NFSS)

See Section 5.3.2.1.1.1.

5.3.2.1.3.2 Monitor, Controller, and Archiver Capability (MCA_CAP)

See Section 5.3.2.1.1.2

5.3.2.1.3.3 Internal Network Capability (IN_CAP)

See Section 5.3.2.1.2.3.

5.3.2.1.3.4 Test-Plan Description Capability (TPD_CAP)

See Section 5.3.2.1.1.3.

5.3.2.2 System Capability Relationships

Table 12 summarizes the relationships between the NFSE System capabilities and the modes of the system. An X indicates that a capability exists. In Modes B and C the system can operate with or without the Internal Network.

Table 12: NFSE System capabilities associated with various system modes.

	Mode				
Name of Capability	A	B	B	C	C
NFSS	X	X	X	X	X
MCA_CAP	X	X	X	X	X
TPD_CAP	X	X	X	X	X
IN_CAP			X		X

5.3.2.3 External Interface Requirement

Figure 4 identifies and names the external interfaces for the NFSE System. There is also an operator interface for monitoring and controlling the NFSE System; however, we treat that as an internal interface, which is described in a later section.

5.3.2.3.1 GPS Interface

The GPS interface shall consist of an RS232 serial port through which the external system shall provide precise Greenwich Mean Time (GMT) and 1 ppm (pulse per millisecond) time intervals for use by the NFSE System.

5.3.2.3.2 Modem Control Interface via NOSC BLC

TBD

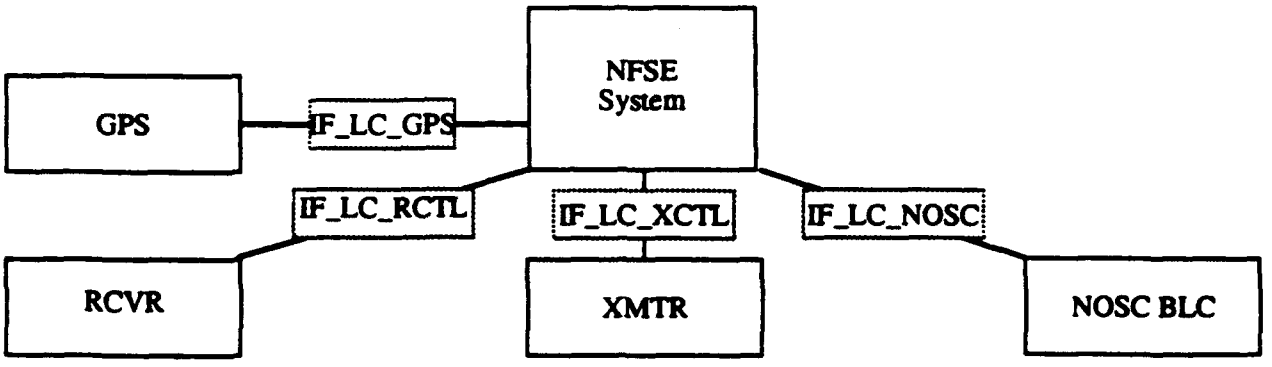


Figure 4 NFSE System external interfaces.

5.3.2.3.3 Modem Data Interface via NOSC BLC

TBD

5.3.2.3.4 Transmitter Control Interface

The transmitter control interface shall allow remote control of all transmitter functions controllable from the front panel, including frequency selection.

5.3.2.3.5 Receiver Control Interface

The receiver control interface shall allow remote control of all receiver functions controllable from the front panel, including frequency selection.

5.4 Quality Assurance Provisions

TBD

5.5 Preparation for Delivery

TBD

5.6 Notes

1. An asynchronous mode of operation for the NFSS was considered but rejected for the following reason. Owing to the unreliability of HF communication links in a threat environment, there is no way to guarantee that, in a timely manner, all nodes in the network could be made aware of an asynchronous request to perform the Network Frequency Selection Service.

5.7 Revisions to Section 5

None.

6. SYSTEM DESIGN

6.1 Scope

6.1.1 Identification

Section 6 provides the equivalent of the System/Segment Design Document (SSDD) for the Network Frequency Selection Experiment (NFSE) System. This design derives from the system requirements, which are given in Section 5.

6.1.2 System Overview

See Section 5.1.2.

6.1.3 Section Overview

This section describes the NFSE system design and its operational and support environments. It describes the organization of the NFSE system as composed of Computer Software Configuration Items (CSCIs) and manual operations. There are no Hardware Configuration Items (HWCIs) included in this system.

6.2 Referenced Documents

See Section 3.

6.3 Operational Concepts

6.3.1 Mission

6.3.1.1 User Needs

The need for a network frequency selection service is documented in [1], which contains the following:

> "The present HF communication systems on most Navy platforms are manually operated. They require retuning of receivers, transmitters, filters and multicouplers every time the frequency is changed, thus requiring a significant amount of labor to operate the system. These changes are required by changes in operating locations/scenarios and changes in sky-wave HF propagation at different times of day. This manual operation takes many minutes to hours, requires a relatively high operator skill level, and is a source of errors causing low system availability."

Subsequently, reference [1] goes on to describe the following effort to address these needs:

> "This effort will incorporate HF networking techniques, and demonstrate automatic frequency selection, network establishment, and connectivity maintenance. This task will examine the dynamics of frequency selection, including: acquiring and distribut-

ing link performance data for all links and frequencies: identifying a "best" frequency or frequencies; distributing that information among all nodes; initiating operation on the selected frequency; and, updating the entire process to maintain good network operation."

6.3.1.2 Primary Mission

The primary mission of the system is to demonstrate a technique for selecting an operating frequency for a single-channel HF network.

6.3.2 Operational Environment

The intended operational environment for the NFSE system is the CSS Testbed, which is described in Section 6.3.3.2.1.

6.3.3 Support Environment

6.3.3.1 Support Concept

This subsection describes the support environment for the operational system during the Production and Deployment phase of the system life cycle.

NRL shall be responsible for hardware and software support for two NFSE System nodes while these nodes are in the development phase. After the NFSE System nodes have been tested in-lab at NRL and have operated properly, the NFSE System software shall be ported to NRaD hardware in preparation for field tests. NRL shall continue to supply maintenance support for the NFSE System software throughout the field test phase. NCCOSC-NRaD shall supply hardware support for the field-test version of the NFSE System.

NRL shall purchase two VME-based multi-processor systems to be used for NFSE System nodes. This hardware shall be similar to that used for the UNT demonstrations. In addition, NRL shall supply one Sun 3 workstation for use in developing the NFSE System software. The latter shall be upgraded to a Sun Sparcstation in 1993.

Software support for the operational system shall consist of the items shown in Table 13. The system console shall run the Unix operating system. The vxWorks software has two components. One of these runs on the console (vxWorks host) and the other runs on the target MVME135 card. The latter includes the vxWorks real-time operating system. The MVME333 boards are supplied with a monitor program, which supports the use of that board.

Table 13: Support software for the operational system

Name	Description
vxWorks	Real-time software environment
Unix	Operating system for the system console
MVME333Bug	Monitor program for MVME333 board

6.3.3.1.1 Government and Contractor Support

NRaD shall be responsible for providing support for the use of the CSS testbed. (See Section 6.3.3.2.1).

NRL may use contractors, as needed, to help prepare for and carry out the field tests.

6.3.3.2 Support Facilities

6.3.3.2.1 CSS Testbed at NCCOSC-NRaD

The Communication Support System (CSS) Testbed at NCCOSC-NRaD shall be the principal site for the NFSE field demonstrations and tests. This testbed consists of five permanent sites, which are shown in Figure 5. NCCOSC-NRaD shall supply the hardware required to conduct the field tests.

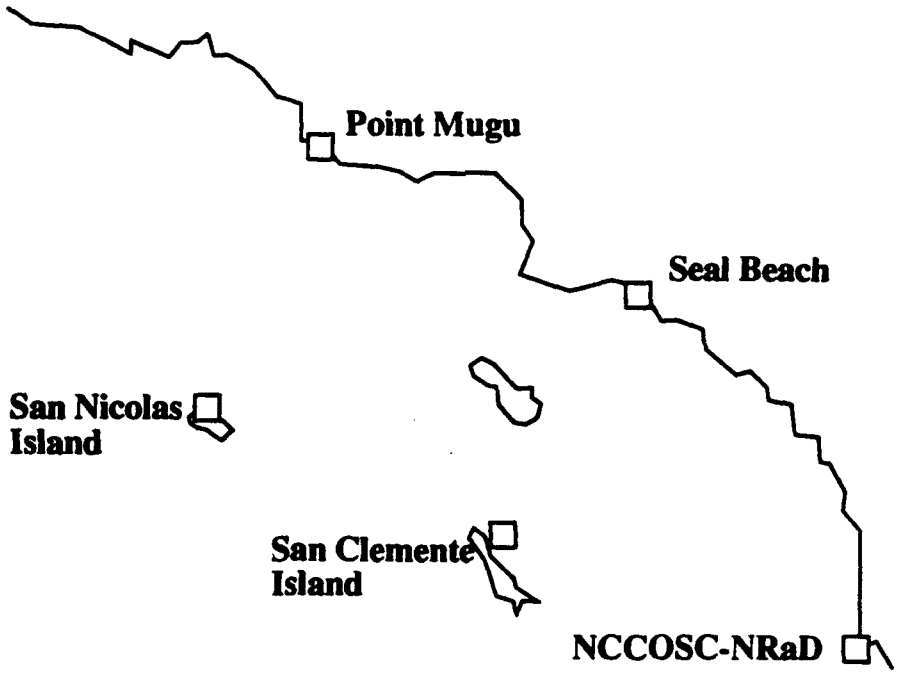


Figure 5 NCCOSC-NRaD CSS Testbed - fixed sites.

6.3.3.3 Supply System

Responsibility for supplying hardware for the NFSE system rests with NCCOSC-NRaD or the organization that they delegate.

6.3.3.4 Government Agencies

NRL shall be responsible for developing and testing the NFSE System. NCCOSC-NRad shall be responsible for the external environment of the NFSE System, as shown in Figure 1.

6.3.4 System Architecture

The top-level components of the NFSE system design and its environment are shown in Figure 6. The NFSE system consists of the following CSCIs: Network Controller (NC), Link Controller (LC), and System Controller (SC). The operator interacts directly with the System Controller (SC) and indirectly, through the SC, with the Network Controller (NC) and Link Controller (LC). The Network Controller and Link Controller also interface to each other.

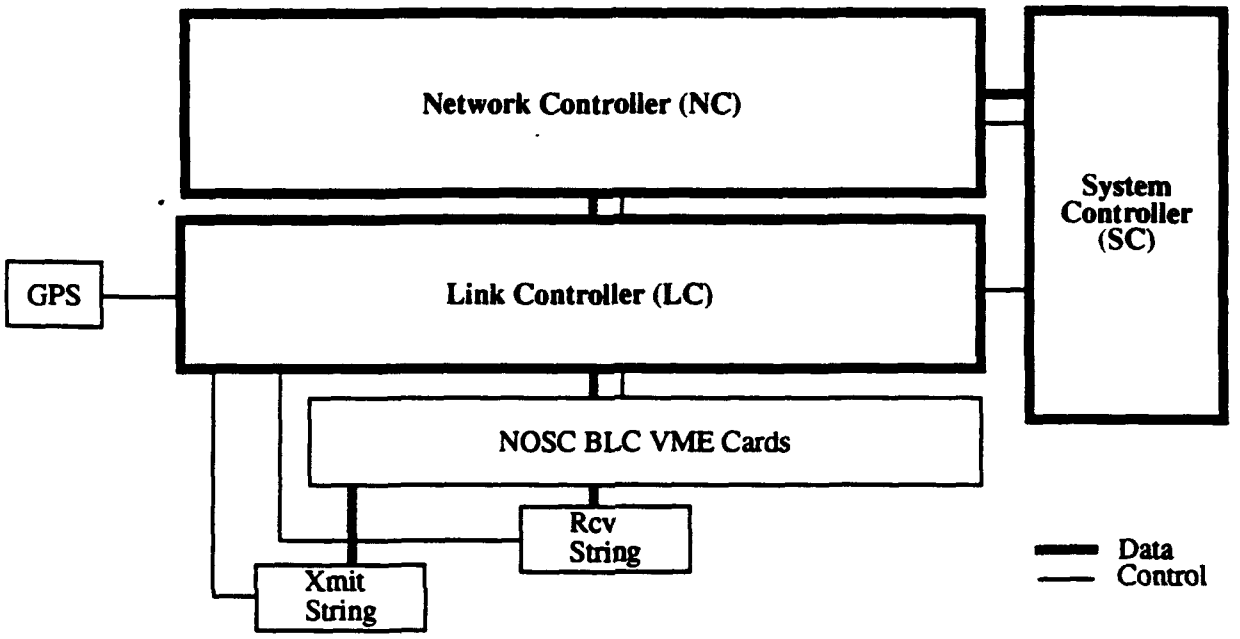


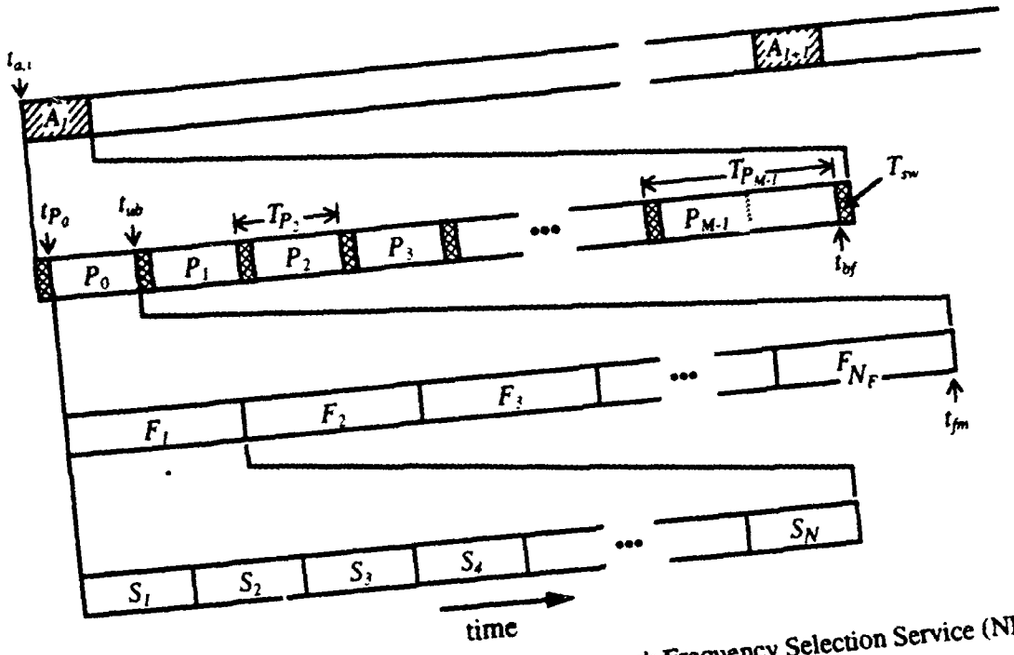
Figure 6 System architecture showing CSCI's and external interfaces.

6.4 System Design

Figure 3 shows an example top-level timing diagram for the NFSE System. Figure 7 provides the next level of detail for the system timing structure.

Time is divided into periods in which the NFSS is active and periods when the NFSS is inactive. During an active cycle, time is divided into M periods of operation at different test frequencies. These frequencies are selected from the sequential collection $\{f_j\}_{sc}$ and stored in the array $fa[]$. Details of this selection process are given in Section 10.1.4.10.1.2.1.

For operation at each frequency test cycle, except the last one, time is divided into N_F frames. The last frequency test cycle is twice as long and contains $2N_F$ frames, Each frame is further divided into N time slots. During the period of operation at a given test frequency, the network goes through the following phases: 1) probing to determine connectivities, 2) network structuring to support the routing of traffic, 3) dissemination of connectivity information by relaying, and finally, 4) evaluation of the suitability of the test frequency.



A_i *i*th scheduled active cycle of the Network Frequency Selection Service (NFSS).

Notes:

A_i - *i*th scheduled active cycle of the NFSS
P_j - NFSS frequency test cycle *j* (using frequency stored in *fa[j]*)
F_1 (Frame 1) - probe, ack/nak probes
F_2 (Frame 2) - ack/nak probes, cluster formation and linkage
F_3 through F_{N_F} - relay connectivity info for current test frequency
N_F
T_{P_j} - Period of time during which the network is operating with the test frequency stored in *fa[j]*.
t_{P_j} - start of the *j*th frequency test cycle P_j
$t_{a,i}$ - start of the *i*th scheduled active cycle of the NFSS
t_{fm} - time of computation of frequency selection metric (t_{fm} precedes t_{ub} at end of P_0)
t_{ub} - time to begin "flood" updates of best frequency and value of metric
t_{bf} - time to switch to best frequency
T_{sw} - delay to allow for completion of frequency switching.

Figure 7 System timing structure

Probing occurs in frame 1. Probes that are received are acknowledged by the receiving node (ack) and probes that are not received are so indicated by having each non-receiving node send out a negative acknowledgment (nak). The process of sending these ack/nak notices occurs over frames 1 and 2.

Based on this connectivity database, the network proceeds to organize itself, during frame 2, into the Linked Cluster Architecture (LCA), as described in [3]. The LCA provides the infrastructure to support multihop communication, which is needed to disseminate the connectivity information throughout the network. Our design allocates a fixed length of time for the network to disseminate connectivity information throughout the network. This time extends from the start of frame 3 to the end of frame N_F. At the end of

frame N_F, each node evaluates the test frequency based on a metric that is described in Section 10.1.4.10.1.2.4.

At the end of the testing period for the first frequency, the network begins the process of "flooding" to all nodes information about the "best" frequency and the value of the metric for that frequency The value of the flooded metric can be thought of as a measure of the strength of the "wave" for that particular flood of messages. As better frequencies are found, a new, stronger wave of "flooding" occurs which extinguishes any previous waves that it encounters. In order that there is sufficient time to complete the flooding process for the case where the best frequency is also the last one tested, the design allows for twice the number of transmission frames for the last test frequency.

6.4.1 HWCI Identification

No HWCIs are included in this system.

6.4.2 CSCI Identification

The NFSE System is composed of three CSCIs, which are called the Network Controller, the Link Controller, and the System Controller.

6.4.2.1 Network Controller (NC) CSCI

6.4.2.1.1 Purpose

The NC implements those functions of the NFSE System that would normally be allocated to the Network Layer in an ISO system architecture. It implements the network frequency selection service and the network communication service requirements given in Section 5.3.2.1. The NC design should place emphasis on the reuse of hardware and software from NRL's UNT project.

6.4.2.1.2 System Capabilities Addressed by the Network Controller

The Network Controller implements the Network Frequency Selection Service (NFSS) and Internal Network Capability (IN_CAP) requirements of the NFSE System. The Network Controller also supports the monitoring capability included in the Monitor, Controller, and Archiver Capability (MCA_CAP) of the NFSE System by providing network status information to the System Controller.

6.4.2.1.3 External System Interfaces Implemented by the NC

The NC does not have any external NFSE System interfaces.

6.4.2.2 Link Controller (LC) CSCI

6.4.2.2.1 Purpose

The LC implements those functions of the NFSE System that would normally be allocated to the Link Layer in an ISO system architecture.

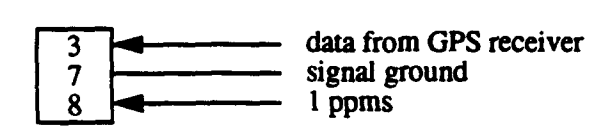
6.4.2.2.2 System Capabilities Addressed by the Link Controller

The LC implements the link level services required by the Network Controller in order for the latter to do its job.

6.4.2.2.3 External System Interfaces Implemented by the LC

6.4.2.2.3.1 GPS Interface (IF_LC_GPS)

The GPS physical interface is an RS232 port, which is configured as shown in Figure 8.



Notes:

1. The GPS port shall be configured as a DTE.
2. Port parameters: 9600 bps, 8 data bits, 1 stop bit, no parity, asynchronous.
3. RS232 voltage levels on pin 3.
4. 0 to 10 volts on pin 8 (into 50 ohm load).
5. Time output on pin 3 to begin within 0.1 millisecond of the UTC 1 sec rollover.

Figure 8 GPS interface.

The serial format for inputting timing information is shown in Table 14. This format corresponds to that of the GPS receiver used in the UNT tests.[14] Bit 0 of byte 1 is the first to be transmitted.

Table 14: GPS serial time-of-day input format.

Byte #	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
1	s1	s2	s4	s8	s10	s20	s40	0
2	m1	m2	m4	m8	m10	m20	m40	
3	h1	h2	h4	h8	h10	h20	0	0
4	d1	d2	d4	d8	d10	d20	d40	d80
5	d100	d200	0	0	0	0	0	0

6.4.2.2.3.2 Modem Interface via NOSC BLC (IF_LC_NOSC)

TBD

6.4.2.2.3.3 Transmitter Control Interface (IF_LC_XCTL)

Initial plans are based on the use of the Harris RT-1446 transceivers, which were used in the UNT demonstrations. The UNT version of this transceiver does not have remote control capability. However, a

remote control interface board (Harris RF-362) can be purchased for the UNT transceivers for $295 each. If this transmitter configuration is used, Harris will supply the specifications for the transmitter control interface.

6.4.2.2.3.4 Receiver Control Interface (IF_LC_RCTL)

The receiver remote control interface is described in [15].

6.4.2.2.4 Design Constraints

The LC design should place emphasis on the reuse of hardware and software from NRL's UNT project.

6.4.2.3 System Controller (SC) CSCI

6.4.2.3.1 Purpose

The primary purpose of the SC is to provide a user interface into the NFSE system. The SC implements the requirements for monitoring, controlling, and data archiving for the NFSE system as specified in Section 5.3.2.1. The SC design should place emphasis on the reuse of hardware and software from NRL's UNT project.

6.4.2.3.2 System Capabilities Addressed by the System Controller

The System Controller implements the Monitor, Controller, and Archiver Capability (MCA_CAP) and the Test-Plan Description Capability (TPD_CAP) of the NFSE System.

6.4.2.3.3 External Interfaces

Although the SC does have an operator interface, we treat this as an internal interface, which is described in Section 11.3.7.

6.4.3 Manual Operations Identification

The Manual Operations for the NFSE System are grouped into the following four categories: test description, system startup, system monitoring and maintenance, and system shutdown. These operations are described below.

6.4.3.1 Describe Test (SET_TEST)

TBD

6.4.3.2 Start System (SYS_START)

TBD

6.4.3.3 Monitor and Maintain System (SYS_RUN)

TBD

6.4.3.4 Shutdown System (SYS_STOP)

TBD

6.4.4 Internal Interfaces

The top-level, internal interfaces of the NFSE system are shown in Figure 9. In addition to the real interfaces, we also show the virtual (peer-to-peer) interfaces. The latter interfaces connect peer layers on separate nodes.

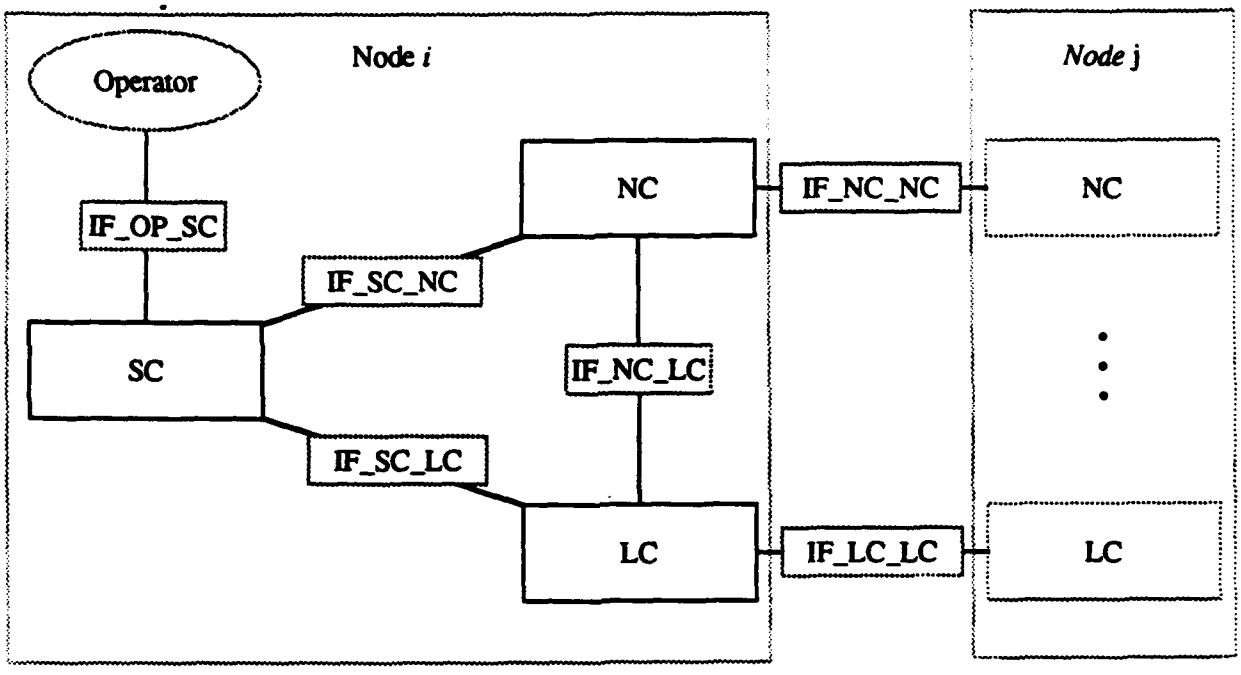


Figure 9 NFSE system internal interfaces.

6.4.4.1 HWCI-to HWCI Interfaces

None.

6.4.4.2 HWCI-to-CSCI Interfaces

None.

6.4.4.3 CSCI-to-CSCI Interfaces

All of the internal interfaces shown in Figure 9 represent CSCI-to-CSCI interfaces. A brief description of each of these interfaces follows.

6.4.4.3.1 IF_OP_SC

IF_OP_SC is the interface between the NFSE System Operator and the System Controller CSCI. Physically, this interface is the keyboard, screen, and mouse that belong to the System console. The software interface is accessed through the Unix operating system of the System console. The requirements for this interface are given in section 9.3.3.

6.4.4.3.2 IF_SC_NC

The workstation, which serves as the System console, interfaces to the Network Controller through the interface IF_SC_NC. Physically, this interface is an ethernet connection. The requirements for this interface are given in section 9.3.5.

6.4.4.3.3 IF_SC_LC

The System console also provides an interface for direct connection to the Link Controller through the IF_SC_LC. Physically, this interface is an RS232 connection. The requirements for this interface are given in section 9.3.4.

6.4.4.3.4 IF_NC_LC

The interface between the Network Controller and the Link Controller at a node is called the IF_NC_LC interface. Physically, this interface is made up of the VME backplane and the MVME224-1 shared memory board within the BLC chassis (see figure 1). The requirements for this interface are given in section 9.3.2.

6.4.4.3.5 IF_NC_NC

There are two peer level interfaces in the NFSE System. These interfaces deal with communication between CSCIs on different nodes. The peer level interface between network controllers on different nodes is called the IF_NC_NC interface. The requirements for this interface are given in section 9.3.6.

6.4.4.3.6 IF_LC_LC

The other peer level interface in the NFSE System is between link controllers on different nodes. This is called the IF_LC_LC interface. Its requirements are given in section 9.3.7

6.5 Processing Resources

The processing resources for the NFSE System are those shown in figure 1. These resources are described below.

6.5.1 System Console (CONSOLE)

The console is a Sun Sparcstation with associated hard disk and tape drive. At least 50 megabytes of storage must be available for use by the NFSE System. The tape drive is needed to archive the test results.

6.5.2 Non-Real-time Network Controller Processor (NC_NRT)

The Non-Real-time NC Processor (Proc #0) must be an MVME135, MVME135A, or suitable substitute.

6.5.3 Real-time Network Controller Processor (NC_RT)

The Real-time NC Processor (Proc #1) must be an MVME135, MVME135A, or suitable substitute.

6.5.4 Shared Memory (M224NRL)

At least 4 megabytes of shared memory must be available to the NFSE System's VME-based hardware. A Motorola MVME224-1 or suitable substitute is required.

6.5.5 Ethernet Controller (LANCNTR)

A VxWorks-compatible ethernet controller is needed to connect the NFSE System's VME chassis to the System's console.

6.5.6 Link Controller Processor (M333NRL)

A Motorola MVME333 or suitable substitute is required by the NFSE System's Link Controller software.

6.6 Quality Factor Compliance

TBD

6.7 Requirements Traceability

Table 15 summarizes the system capabilities addressed by each of the HWCIs, CSCIs, and Manual Operations of the NFSE System.

Table 15: Requirements Traceability Matrix

	NFSS	MCA_CAP	IN_CAP	TPD_CAP
SC		X		X
NC	X	X	X	
LC	X	X	X	
SET_TEST				X
SYS_START	X	X	X	
SYS_STOP	X	X	X	

Table 15: Requirements Traceability Matrix

	NFSS	MCA_CAP	IN_CAP	TPD_CAP
SYS_RUN	X	X	X	

6.8 Notes

None.

6.9 Revisions to Section 6

None.

7. SOFTWARE DEVELOPMENT PLAN

Section 7 provides the equivalent of a 2167A Software Development Plan (SDP) for the NFSE System.

7.1 Scope

7.1.1 Identification

This SDP is for the NFSE System and is applicable to all CSCIs identified in Section 6.4.2.

7.1.2 System Overview

See section 5.1.2 on page 13.

7.1.3 Section Overview

This section describes NRL's plans for conducting software development for the NFSE System.

7.2 Referenced Documents

See Section 3.

7.3 Software Development Management

7.3.1 Project Organization and Resources

7.3.1.1 NRL Facilities to be used

NRL facilities to be used include office spaces, computer resources, and the Distributed Simulation and Prototyping Testbed (DSPT). These facilities are presently located at NRL in Building 26.

7.3.1.2 Government Furnished Equipment, Software, and Services

The hardware, shown in Table 16, and the software, shown in Table 17, shall be obtained with project funds.

Table 16: Hardware Items to be Obtained

Item	Description	Qty.	Ordered	Received
MVME945B-1	VME Benchtop Enclosures	2	Yes	Yes
MVME135A	CPU (4 Mbyte DRAM)	5	Yes	Yes
MVME224A-1	Memory (4 Mbyte DRAM)	2	Yes	Yes
MVME333-2	Serial I/O Module	2	Yes	Yes

Table 16: Hardware Items to be Obtained

Item	Description	Qty.	Ordered	Received
MVME705A	Transition Board for 333	2	Yes	Yes
ENP-10L	Ethernet Controllers	2	Yes	Yes
Harris RF-362	Remote Controller board	7	No	No

Table 17: Software Items to be Obtained

Item	Description	Qty.	Ordered	Received
vxWorks	Real-time software environment	1	Yes	Yes

7.3.1.3 Organization Structure

Code 5521 personnel at NRL shall be responsible for the development of the NFSE System. This code's position in the NRL organizational structure appears in Figure 10.

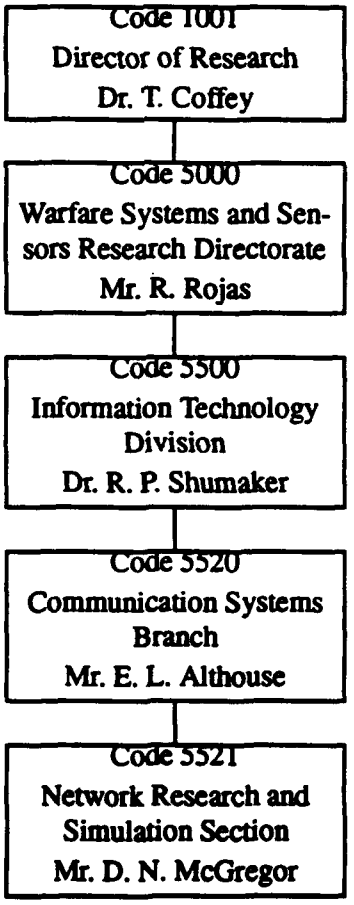


Figure 10 Code 5521's position in NRL organizational structure.

7.3.1.4 Personnel

The manager for the NFSE System Development Project, hereafter to be referred to as the Project, is Mr. Dennis N. McGregor, Section Head for NRL Code 5521. The Technical Leader for the Project is Dr. Dennis J. Baker, a member of this Section.

7.3.2 Schedule and Milestones

Table 18 shows the schedule for performing the tasks associated with NFSE project. Also displayed are the milestones, which specify when the deliverables shown in table 19 are due.

Table 18: Schedules and milestones

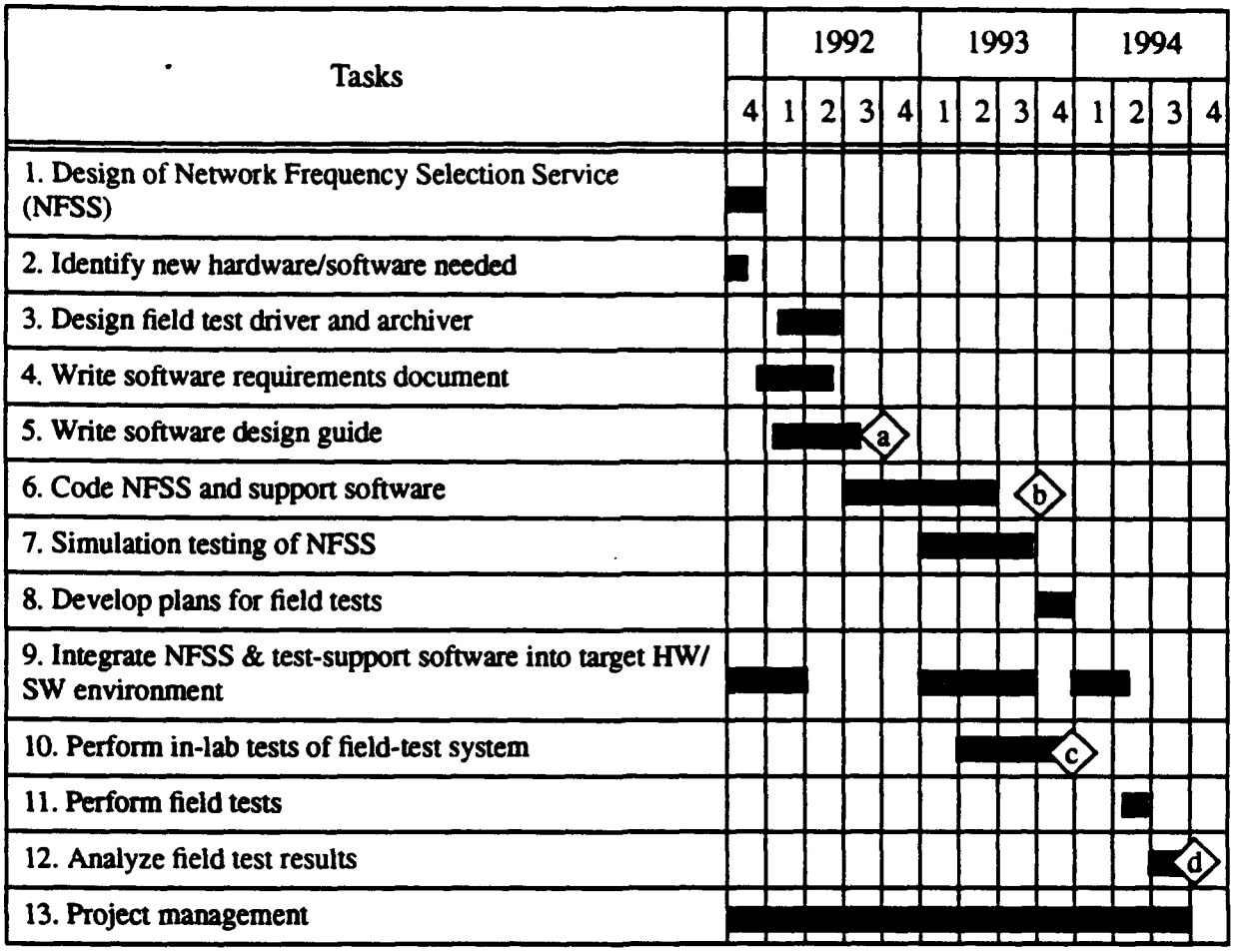


Table 19: Deliverables

Milestone	Deliverable
a	Software development documentation
b	C++/C code for NFSS
c	Operational node
d	Final report

7.3.2.1 Activities

The set of tasks necessary to complete the NFSE project are listed below.

Task 1 - *Design of Frequency Selection Service (NFSS)* - This task is to develop an implementable algorithm by which N nodes can select the best operating frequency from a given set of allocated HF frequencies. The algorithm must be robust with respect to communication connectivity changes, irrespective of the causes of these changes. The task must address the following: (1) the measurements that must be made to evaluate link qualities at a given frequency, (2) the protocol for disseminating this link quality information throughout the network, (3) the development of a metric for selecting the best frequency on a network-wide basis, and (4) and a protocol for installing the chosen frequency throughout the network. Both centralized and distributed techniques for implementing the NFSS shall be considered. The output of this task shall be a software specification of the NFSS.

Task 2 - *Identify new hardware/software needed* - The objective of this task is to identify any special hardware and software needed to implement the NFSS. For example, it is almost certain that the NFSS will require computer control of both the transmitter and receiver. The resource requirements specification will also give estimates of the cpu and memory requirements of the NFSS, as well as specifying any timing constraints that must be met. This task will also identify any special hardware/software development tools that are needed to fully implement the NFSE System.

Task 3 - *Design Field Test Driver and Archiver* - Additional software is needed for the following: 1) to provide traffic loads on the network during the field demonstrations, 2) to monitor the operation of the NFSS during the field tests, and 3) to archive the results of its operation for more detailed post-analysis of its performance. The basic design for accomplishing these functions shall be the focus of this task.

Task 4 - *Write Software Requirements Document* - The purpose of this task is to generate specifications for the NFSS software and any supporting software. The latter may include test drivers, data archiver, and test monitor display software.

Task 5 - *Write software design guide* - The purpose of this task is to provide a software design document, which will serve as the basis for software coding.

Task 6 - *Code the NFSS and test-support software* - The purpose of this task is to use the documentation and specifications of Tasks 4 and 5 as the guide for developing the requisite NFSE System software for the target hardware. The initial effort on this task shall focus on construction of the software development environment that shall be used during the subsequent code development.

Task 7 - *Simulation Testing of the NFSS* - This task is to perform a simulation of the NFSS design and run appropriate tests so as to verify that the design is not flawed.

Task 8 - *Develop Plans for Field Tests* - The purpose of this task is to generate a document describing the NFSE System as it will be performed in the field. This document will describe the test objectives, test procedures, data collection requirements, and data analysis procedures. If any

special monitoring tools are needed to show the operation of the NFSS during the actual field demonstrations, these tools shall also be described in the test plans.

Task 9 - *Integrate NFSS & Test-Support Software into Target Hardware/Software Environment* - This task provides for the integration of the NFSS and support software into the field testbed hardware/software environment. The task is broken into three phases, as indicated in the schedule of Table 18. During the first phase, the goal is to reconstruct an HF network node as it existed in NRL's UNT demonstration. The NFSE System node shall be obtained by enhancing the capabilities of the UNT node. During the second phase of this task, the software for the new Network Frequency Selection Service and related services shall be integrated into the target hardware. During the third phase of this task, the NFSE System node shall be integrated with the NOSC BLC.

Task 10 - *Perform in-lab Tests of Field-Test System* - The objective of this task is to perform back-to-back testing in the laboratory of at least two, fully configured nodes.

Task 11 - *Perform Field Tests* - The actual demonstration or field tests would be accomplished over a period of about one month, starting with only two or three nodes and culminating with tests using five or more nodes.

Task 12 - *Analyze Field-Test Results* - The purpose of this task is to analyze the data from the field tests and describe the results in a final report.

Task 13 - *Project Management* - This task covers manpower resource scheduling, coordination with sponsor, managing project finances, providing briefings, obtaining contractual support as required, and other necessary project management functions.

7.3.2.2 Activity Network

Not available.

7.3.2.3 Source Identification

This subsection identifies and describes the sources of the required resources (software, firmware, and hardware) for the software development effort for the NFSE System. We outline a plan for obtaining the required resources and indicate the need date. If the availability of a particular item is known, we also give that information.

The resources that are needed can be grouped into those needed for each of the following three phases: 1) NFSS development, 2) hardware/software integration performed at NRL, and 3) final hardware/software integration for field-test system. The resources needed for Phase 1 are listed in tables 20 and 21. Those required in Phase 2 are given in tables 22 and 23. And those that are needed in Phase 3 are given in tables 24 and 25.

Table 20: Hardware Resources needed for NFSS Development

Item	Description	Source	Qty.	Date Needed	Availability
MVME945B-1	Chassis	Motorola	2	Start of Task 9	Received
MVME135A	CPU	Motorola	5	Start of Task 9	Received
MVME224A-1	Memory	Motorola	2	Start of Task 9	Received
MVME333-2	Serial I/O	Motorola	2	Start of Task 9	Received
MVME705A	Transition Board	Motorola	2	Start of Task 9	Received
ENP-10L	Ethernet Board	Rockwell	2	Start of Task 9	Received
Sun 3	Workstation	NRL	1	Start of Project	Available
Sun Sparcstation	Workstation	NRL	1	TBD[a]	Ordered

a. The Sun Sparcstation will be needed when we receive the version of vxWorks that supports C++. This is expected to occur around April of 1993.

Table 21: Software Resources needed for NFSS Development

Item	Description	Source	Qty.	Date Needed	Availability
vxWorks	Real-time software	Wind River	1	Start of Task 9	Received
vxWorks (for C++)	Real-time software	Wind River	1	ASAP	March 1993

Table 22: Hardware Resources needed for Field-Test Node Development at NRL

Item	Description	Source	Qty.	Date Needed	Availability
TBD	TBD	TBD	TBD	1 year prior to start of field tests	TBD

Table 23: Software Resources needed for Field-Test Node Development at NRL

Item	Description	Source	Qty.	Date Needed	Availability
TBD	TBD	TBD	TBD	1 year prior to start of field tests	TBD

Table 24: Hardware Resources needed for final integration with Field-Test System

Item	Description	Source	Qty.	Date Needed	Availability
TBD	TBD	TBD	TBD	1 year prior to start of field tests	TBD

Table 25: Software Resources needed for final integration with Field-Test System

Item	Description	Source	Qty.	Date Needed	Availability
TBD	TBD	TBD	TBD	Need access to HW/SW 6 mo. prior to start of field tests	TBD (need access to HW/SW)

7.3.3 Risk Management

For the most part, there is very low risk during the first two years of the project to develop and test the Network Frequency Selection Service. Most of the high risk areas occur in the third year. Table 26 lists the major risk areas. Each risk area is ranked as low, medium, or high risk with regards to technical, cost schedule, and network (risk caused when problems with one task have an adverse impact on a large number of other tasks) factors. The following paragraphs elaborate on these risk areas.

Table 26: Assessment of Risk Areas

Risk Area	Technical Risk	Cost Risk	Schedule Risk	Network Risk	Overall Risk
Software Development Language	Medium	Low	Low	Medium	Medium
Multi-Contractor, Multi-Site	High	High	High	High	High
Field-Test Resources Availability	n.a.	High	High	High	High

7.3.3.1 Software Language Development

The high-level language C++ has been selected for most of the software for the NFSE System. This was the language used to implement the Network Controller for the UNT/NRL tests. It is a risk area because C++ is not directly supported by the vxWorks real-time software. Nevertheless, it is felt that this is not a high risk area because, if needs be, C code can be obtained from the C++ code, and vxWorks does support C. Two additional steps taken to reduce the risks associated with using C++ were the following: 1) enlist the services of a contractor (Jade Simulations International) that is familiar with porting C++ to

VME systems, and 2) obtain C++ support from vxWorks as soon as it becomes available. In regards the latter, we are seeking to have NRL serve as a beta test site for the soon-to-be-tested C++ support for vxWorks.

7.3.3.2 Multi-Contractor and Multi-Site

During the third year of this project, there is a large amount of closely coordinated work that must be done by NRL and NCCOSC-NRaD workers. This arises because of the need to integrate the NFSE System hardware and software into a single VME chassis that is shared with NCCOSC-NRaD hardware and software. There are many potential problems that can arise as a result of such an integration. Some of these problems could be quite difficult to diagnose and correct unless extremely close cooperation is achieved between the two groups of workers. Adding to the difficulty of achieving the required level of coordination between the two groups is the large geographical separation of the two laboratories. Costs are another consideration in bringing the two groups together for the required length of time. An alternative to transporting one of the groups to the others facility for an extended length of time is to make available to both groups a fully-configured and operating node, which contains both the NRL and NRaD hardware and software. Then both groups could perform tests with a fully integrated system, and integration bugs could be worked out with phone calls among the NRL and NRaD researchers.

7.3.3.3 Field-Test Resources Availability

A critical item for the successful execution of the third year field-test of the NFSS, is the availability of the hardware needed by NRL researchers to construct a complete and functioning node. This includes all of the hardware and software that would exist at a node in the field test. Such hardware includes: transmitters, receivers, modems, GPS receivers, multicouplers, antennas, etc. The second critical need is for NRL researchers to have adequate access to all field test facilities for preliminary testing and debugging.

7.3.3.4 Minimizing the Risks Associated with Field Tests

As noted, all of the high risk areas are associated with field-tests that require the integration of NRaD and NRL hardware and software into a single system. Such an integration effort is not necessary to prove the concepts of a Network Frequency Selection Service. In an effort to avoid these integration costs, the NFSE System is being designed to work with or without a co-resident HAMA network.

7.3.4 Security

No security features shall be included in this initial demonstration of a Network Frequency Selection Service.

7.3.5 Interface with Associated Contractors

Jade Simulations International (JSI) shall provide assistance with code development and simulation of the NFSE System. The point of contact for JSI is Dr. Greg Lomow. Dr. Dennis Baker of NRL shall provide technical direction for the JSI effort. Mr. Dennis McGregor of NRL Code 5521 shall serve as the Contracting Officer's Technical Representative (COTR) on the JSI contract.

7.3.6 Interface with Software IV&V Agents

Not applicable.

7.3.7 Subcontractor Management

Dr. Dennis Baker of NRL Code 5521 shall work directly with JSI contracting personnel on a day to day basis and provide the necessary technical guidance to the subcontractor.

7.3.8 Formal Reviews

NRL shall present progress on the development and testing of the Network Frequency Selection Service at the annual Office of Naval Technology's (ONT) 6.2 Block Reviews.

7.3.9 Software Development Library

The software de velopment library (SDL) consists of all documents, source code, compiled code, change requests, etc. associated with the system. It also consists of all commercial off-the-shelf (COTS) and non-developmental software used for conducting and supporting the project.

Formal tracking and change control shall be imposed on this document and on all source code that makes up the NFSE System. The Unix-based Revision Control System (RCS) shall be used to implement version control for all source code.

7.3.10 Corrective Actions Process

Corrective actions shall be reported to Dr. Dennis Baker of NRL Code 5521 for authorization to make changes.

7.3.11 Problem/Change Report

No special format is required to report problem. However, a request for change should include the following: a description of the proposed change, justification for the proposed change, identification of the person suggesting the change, and date the change request was prepared.

7.4 Software Engineering

7.4.1 Organization and Resources - Software Engineering

7.4.1.1 Organizational Structure - Software Engineering

NRL Code 5521 is responsible for performing all software engineering for the NFSE System. JSI shall provide subcontractor support for this effort as needed.

7.4.1.2 Personnel - Software Engineering

Dr. Dennis Baker shall be in charge of software engineering for the NFSE System development. He shall be assisted in this task, as needed, by JSI personnel.

7.4.1.3 Software Engineering Environment

This subparagraph identifies and describes the plans for establishing and maintaining the resources (software, firmware, and hardware) necessary to performing the software engineering activities.

7.4.1.3.1 Software Items

The following software shall be available for supporting the NFSE System during development. NRL shall procure vxWorks real-time software support for this project. In addition, NRL shall make the following, in-house software, available for support of the NFSE System: C++ compiler/translator (Gnu C++ compiler and/or C-front translator), C++ software development environment (Object Center), commercial simulation software (Sim++), word processor (FrameMaker), editor (Gnu Emacs), software version control (Revision Control System (RCS)), system design support (Turbo-Case).

NRL shall also make available for this project the Distributed Simulation and Prototyping Testbed (DSPT). The DSPT provides an environment for conducting both real-time and non-real-time tests of communication systems software. It does this by emulating the software environment in which the communication software is to be immersed. The DSPT provides software interfaces to RF transmission hardware, which emulate the target system interfaces and which interact via simulated communication channels. It includes models for both HF groundwave and LOS communication channels. The DSPT provides a limited operating system interface. (In the NRL/UNT tests, this limited operating system interface was sufficient to meet all of the operating system demands of the target communication software.) During the development of the NFSE System software, the DSPT shall be the primary tool for conducting in-lab tests of this system. The DSPT software shall be maintained under the RCS system.

7.4.1.3.2 Hardware and Firmware Items

Hardware items needed in support of the software engineering effort are listed in section 7.3.2.3.

7.4.1.3.3 Proprietary Nature and Government Rights

All software developed for the NFSS is the property of the government and shall be delivered without any restrictions on its use.

7.4.1.3.4 Installation, Control, and Maintenance

NRL Code 5521 is responsible for installing, controlling, and maintaining the NFSE System software.

7.4.2 Software Standards and Procedures

7.4.2.1 Software Development Techniques and Methodologies

MIL-STD-2167A is being used as a guide in preparing the documentation for the NFSE System. Object-oriented programming (OOP) is the methodology being used for code development. In addition, an effort is being made to include some of the techniques of formal methods in the software specification phase of the NFSE System development.

7.4.2.2 Software Development Files

Prior to the field tests, NRL shall perform in-lab tests of each CSCI. The results of this activity shall be placed in the Software Development Files (SDF) for that CSCI. Plans for the creation and maintenance of SDFs, their formats and contents, and the procedures for maintaining them, is yet to be determined.

7.4.2.3 Design Standards

Object-oriented design (OOD) is the primary design standard for the NFSE System software.

7.4.2.4 Coding Standards

TBD

7.4.3 Non-developmental Software

See section 7.4.1.3.1.

7.5 Formal Qualification Testing

7.5.1 Organization and Resources - Formal Qualification Testing

TBD

7.5.2 Test Approach/Philosophy

The Network Controller CSCI will be tested separately. The Link Controller CSCI will be tested together with the Network Controller, and finally, the System Controller, Network Controller, and Link Controller will be tested together.

7.5.3 Test Planning Assumptions and Constraints

In-lab testing is being planned under the assumption that no modems, RF equipment, or NRaD BLC hardware or software will be available.

7.6 Software Product Evaluations

7.6.1 Organizational Structure - Software Product Evaluations

See section 7.3.1.3.

7.6.2 Software Product Evaluations Procedures and Tools

TBD

7.7 Software Configuration Management

See section 7.3.9.

7.8 Other Software Development Functions

Not applicable.

7.9 Notes

None.

7.10 Revisions to Section 7

None.

8. SOFTWARE REQUIREMENTS

This section serves as a Software Requirements Specification (SRS) for each CSCI of the NFSE System.

8.1 Network Controller (NC) CSCI

8.1.1 Scope

8.1.1.1 Identification

This section describes the software requirements for the Network Controller (NC), which is a CSCI of the NFSE System.

8.1.1.2 NC Overview

See Section 6.4.2.1.1.

8.1.1.3 Subsection Overview

Section 8.2 specifies the engineering and qualification requirements for the Network Controller CSCI.

8.1.2 Applicable Documents

See Section 3.

8.1.3 Engineering Requirements

8.1.3.1 NC External Interface Requirements

Figure 11 shows the modules with which the NC interacts and identifies the interfaces through which these interactions take place. The specifications of these interfaces are given in section 9.3.

8.1.3.2 NC Capability Requirements

The Network Controller shall implement the following specified capabilities.

8.1.3.2.1 NC-to-NC Communication (NC_COM) Requirement

The NC_COM capability of the NC CSCI shall provide the following communication and communication-related services.

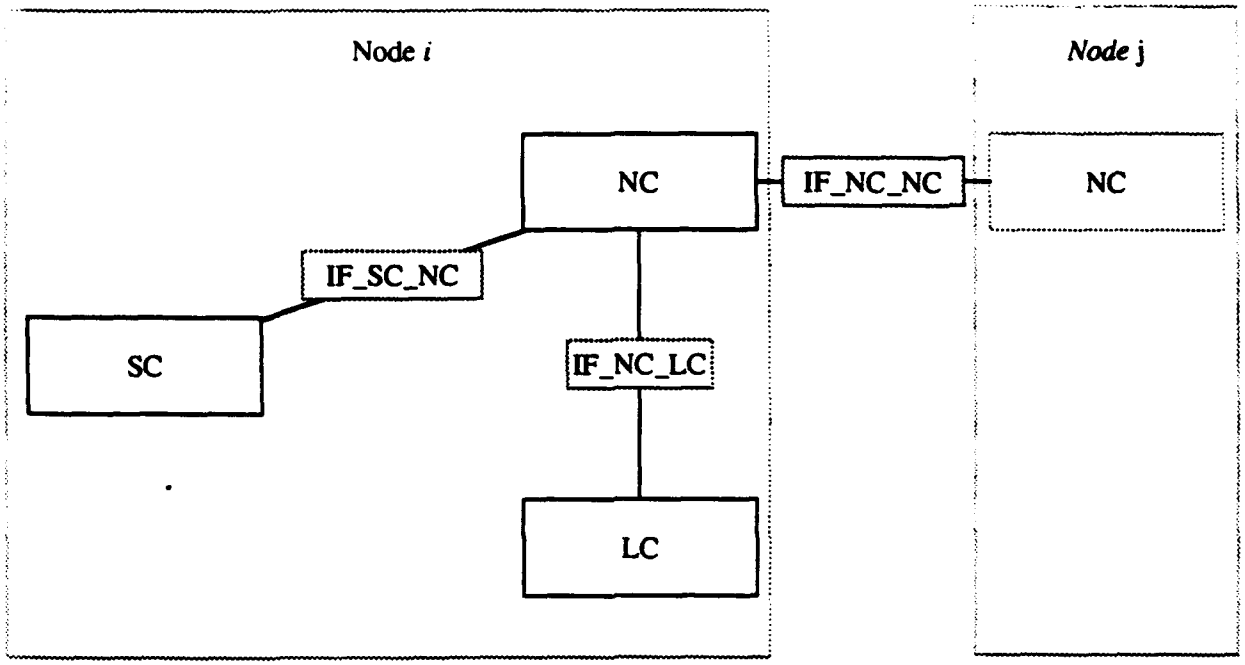


Figure 11 NC external interface requirements.

- *Point-to-point message service* - This service shall provide a means of sending messages from an NC port at one node to the designated destination node and NC port. Once the NC_COM function accepts a message for delivery, it shall deliver the message to the destination node with an average delay that shall not exceed the hop count times the frame duration, where the hop count is the number of relay transmissions needed to reach the destination. Duplicate messages shall be detected and discarded.
- *Broadcast message service* - This service shall provide a means of sending messages from an NC port at one node to a designated port at every other node. Once the NC_COM function accepts a message for delivery, it shall deliver the message to the destination nodes with an average delay that shall not exceed the hop count times the frame duration, where the hop count is the number of relay transmissions needed to reach a destination. Duplicate messages shall be detected and discarded.
- *Local message service* - This service shall provide a means of sending messages from an NC port at one node to the designated port at any node that is directly connected to the source node.
- *Network connectivity learning service* - This service shall provide a measure of both short-term and long-term link qualities for all possible links in the network.
- *Performance monitoring service* - This service shall provide a means for gathering and reporting NC performance data.

A message should not be greater than MAX_USER_DATA. User traffic that exceeds MAX_-USER_DATA should be segmented and reassembled by higher-level protocols.

The NC_COM capability will be obtained by modifying the NC used by NRL in the UNT project.

8.1.3.2.2 Network Frequency Selection Service (NFSS) Requirement

See section 5.3.2.1.1.1.

8.1.3.3 NC Internal Interfaces

Figure 12 shows the interfaces internal to the NC. These interfaces are described in the following subsections.

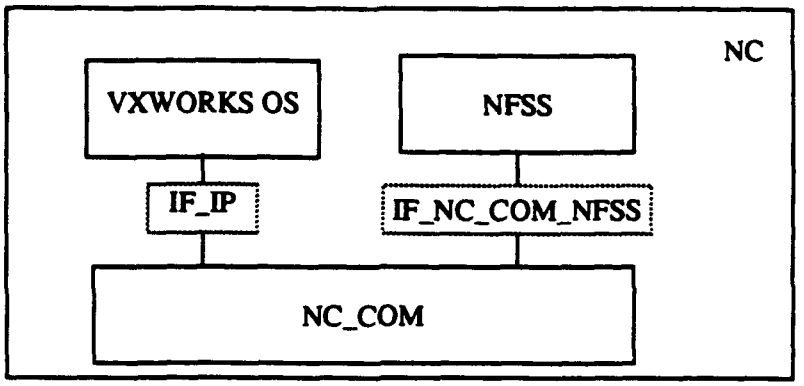


Figure 12 NC internal interfaces

8.1.3.3.1 Internet Protocol (IP) Interface (IF_IP)

The IF_IP interface is the vxWorks network interface driver for the NFSE System's internal network. The requirements for this driver are given in [12].

8.1.3.3.2 NFSS to NC_COM Interface (IF_NC_COM_NFSS)

The IF_NC_COM_NFSS interface connects the Network Frequency Selection Service to the communication services (NC_COM) that it needs to perform its task.

8.1.4 Qualification Requirements

In-lab demonstrations shall be used to verify that the Network Controller CSCI implements the capabilities described in section 8.1.3.2.

8.1.5 Preparation for Delivery

The NFSS source code shall be delivered in hard copy form.

8.1.6 Notes

None.

8.2 Link Controller (LC) CSCI

8.2.1 Scope

8.2.1.1 Identification

This section describes the software requirements for the Link Controller (LC), which is a CSCI of the NFSE System.

8.2.1.2 LC Overview

See Section 6.4.2.2.1.

8.2.1.3 Subsection Overview

Section 8.2 specifies the engineering and qualification requirements for the Link Controller CSCI.

8.2.2 Applicable Documents

See Section 3.

8.2.3 Engineering Requirements

8.2.3.1 LC External Interface Requirements

Figure 13 shows the external modules with which the LC interacts and identifies the interfaces through which these interactions take place. These interfaces are specified in section 9.

8.2.3.2 LC Capability Requirements

8.2.3.2.1 LC-to-LC Communication (LC_COM) Requirement

The LC shall provide communication with other LCs that are within direct communication range and provide the lower level communication services required by the NC. Specific capabilities provided by the LC_COM function are as follows.

- Provide time-synchronous, block-oriented communication service to the NC that allows communication between nodes that are directly connected. (The smallest unit of data exchange between the LC and NC shall be a block.)
- Provide flag indicating to the NC whether a block was received, and if it was received, indicate whether it was received without errors.
- Each block shall be capable of holding M_BLOCKSIZE bytes of data from the NC (not including checksum bytes added by the LC).
- When the NC delivers a block of data to the LC it shall indicate the time at which the block transmission is to occur.

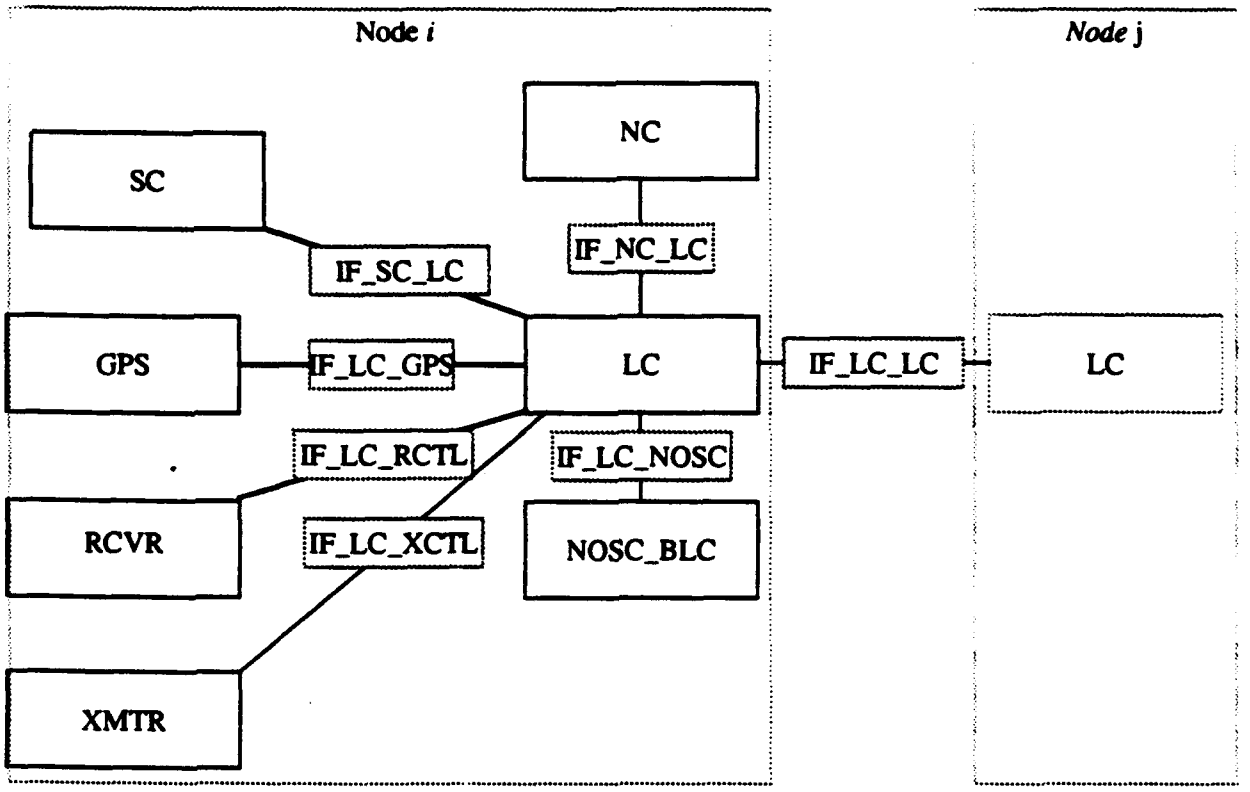


Figure 13 LC external interface requirements.

- An LC shall transmit a block of data within 10 milliseconds of its scheduled transmission time.
- A block received from the modem by the LC shall be delivered to the NC within 270 milliseconds of the reception of the entire block by the LC.
- Blocks shall be delivered to the LC from the NC in time-of-transmission order.

The LC_COM module shall also provide the NC with the capability to exercise the following control over the attached transmitter and receiver.

- Set transmit frequency at given time.
- Set receive frequency at given time.

8.2.3.2.2 Node Timing (ND_TIME) Requirement

The LC shall provide precise time of day and 1 pulse-per-millisecond signals to the NC.

The LC shall schedule the sequencing of events and provide the required data manipulation to drive the RF hardware.

8.2.3.3 LC Internal Interfaces

The internal interfaces between the LC functional capabilities are identified in figure 14.

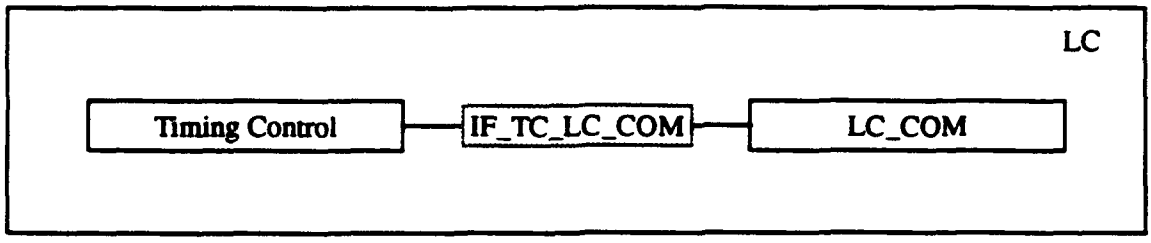


Figure 14 LC internal interface requirements.

8.2.4 Qualification Requirements

In-lab demonstrations shall be used to verify that the Link Controller CSCI implements the capabilities described in section 8.2.3.2.

8.2.5 Preparation for Delivery

Not applicable.

8.2.6 Notes

None.

8.3 System Controller (SC) CSCI

8.3.1 Scope

8.3.1.1 Identification

This section describes the software requirements for the System Controller, which is a CSCI of the NFSE System.

8.3.1.2 CSCI Overview

See Section 6.4.2.3.1.

8.3.1.3 Subsection Overview

Section 8.2 specifies the engineering and qualification requirements for the System Controller CSCI.

8.3.2 Applicable Documents

See Section 3.

8.3.3 Engineering Requirements

8.3.3.1 SC External Interface Requirements

Figure 15 shows the external modules with which the System Controller interacts and identifies the interfaces through which these interactions take place. These interfaces are specified in section 9.

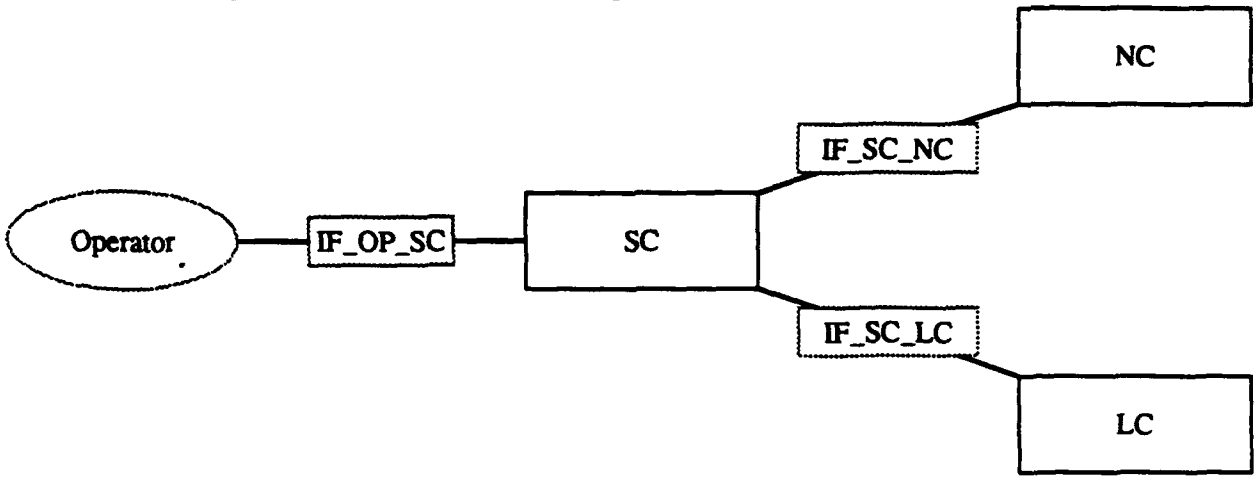


Figure 15 SC external interface requirements.

8.3.3.2 SC Capability Requirements

8.3.3.2.1 System Performance Archiving (SC_ARCHIVE) Requirement

The System Controller shall provide for the logging of NFSE System performance data to disk file.

8.3.3.2.2 System Monitor and Control (SC_MON) Requirement

The System Controller shall provide a system monitoring and control capability that is accessible by the System Operator. The interface to this service shall provide the operator with a means of ascertaining the status of the NFSE System and starting and stopping the system.

8.3.3.3 SC Internal Interfaces

There are no internal interfaces between the System Controller functional capabilities.

8.3.4 Qualification Requirements

In-lab demonstrations shall be used to verify that the System Controller CSCI implements the capabilities described in section 8.3.3.2.

8.3.5 Preparation for Delivery

Not applicable.

8.3.6 Notes

None.

8.4 Revisions to section 8

None.

9. INTERFACE REQUIREMENTS SPECIFICATION

9.1 Scope

9.1.1 Identification

This section serves as the Interface Requirements Specification (IRS) for the NFSE System.

9.1.2 System Overview

See section 5.1.2 on page 13.

9.1.3 Section Overview

This section specifies the requirements for the interfaces between the CSCIs that make up the NFSE System.

9.2 Applicable Documents

See Section 3

9.3 Interface Specification

9.3.1 Interface Diagrams

Figure 16 shows the interface diagram for the NFSE System. The interfaces IF_LC_GPS, IF_LC_RCTL, IF_LC_XCTL, and IF_LC_NOSC connect the NFSE System to its external environment. Those interfaces are specified in section 6.4.2.2.3. This IRS covers the other interfaces shown in figure 16.

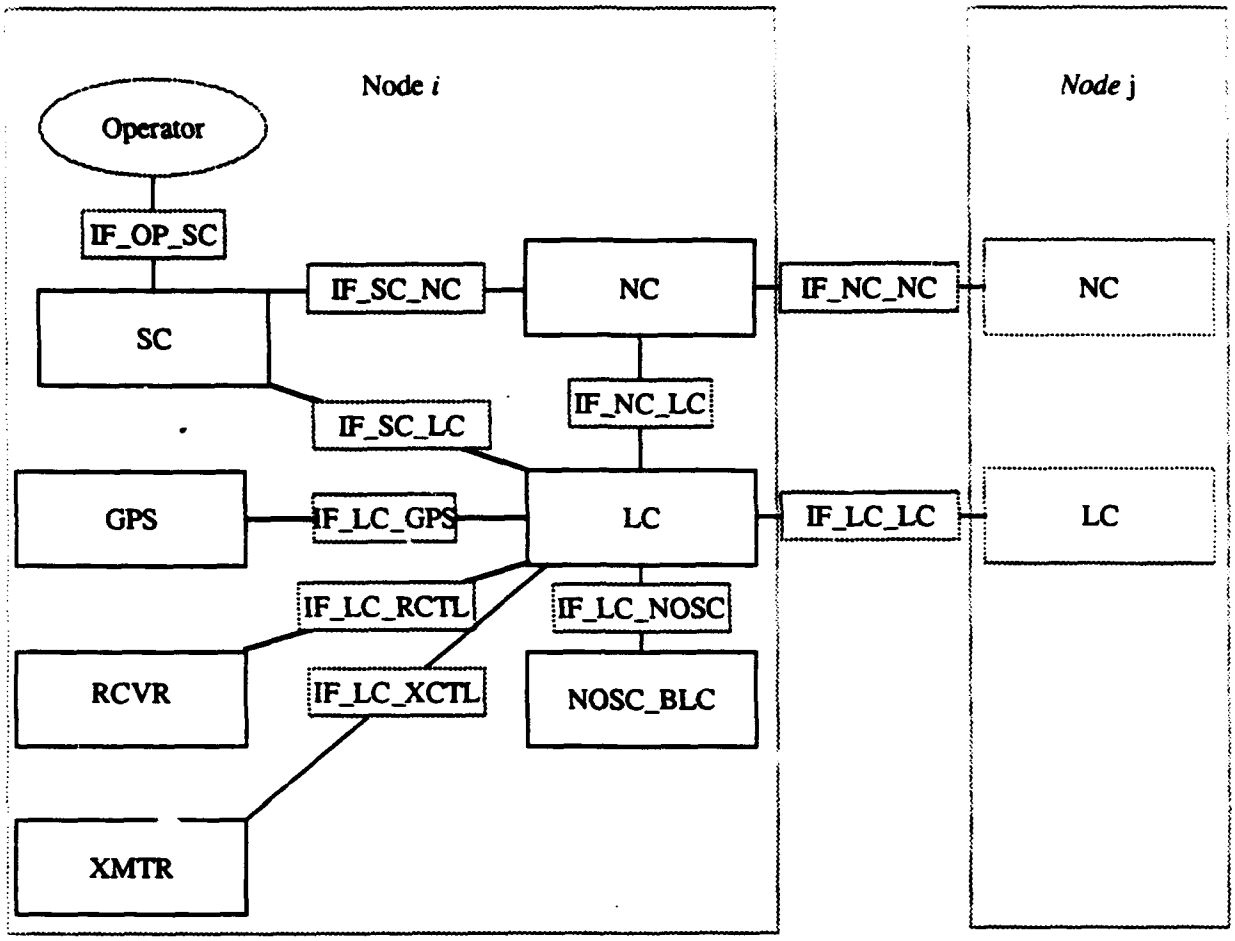


Figure 16 NFSE System interfaces

Table 27 gives a listing of the NFSE System interfaces that are specified in this IRS.

Table 27: Listing of NFSE System interfaces specified in the IRS.

Interface Descriptor	Interface Identifier
LC-to-LC Peer Interface	IF_LC_LC
Network Controller/Link Controller Interface	IF_NC_LC
NC-to-NC Peer Interface	IF_NC_NC
Operator/System Controller Interface	IF_OP_SC
System Controller/Link Controller Interface	IF_SC_LC
System Controller/Network Controller Interface	IF_SC_NC

9.3.2 Network Controller/Link Controller Interface (IF_NC_LC)

9.3.2.1 Interface Requirements

The NC and LC CSCIs execute concurrently. The LC executes on the M333NRL board and interfaces via shared memory board M224NRL to the NC Real-Time processor board NC_RT (Proc #1). The interface requirements for IF_NC_LC are similar to the NC/LC interface described in [10].

All communications between the NC and LC take place via queues and data buffers located on the M224NRL shared memory board. The NC and LC are each responsible for maintaining its own queues and buffers and for the movement of data between its own local memory and shared memory when required. Commands are passed between the NC and LC via command queues which are constantly monitored. When a command from the NC has been detected, the LC immediately (within 1 ms) copies the command parameters to it local memory. The LC performs a validity test on the command to ensure it is valid and will reject it if it is not. Rejected commands are immediately returned to the NC with a suitable error code. The command queue is then reset to "idle" signaling the NC that the command has been processed and the queue slot is again available for use. Commands from the LC to the NC are processed in the same manner by the NC.

9.3.2.2 Data Requirements

The data requirements for the IF_NC_LC interface are described in file common.h, which appears in [9].

9.3.3 Operator/System Controller Interface (IF_OP_SC)

This provides the man-machine interface for the system operator.

9.3.3.1 Interface Requirements

The console screen shall provide the windows shown in figure 17. The windows labeled "NC_RT" and "NC_NRT" connect to the MVME 135 NC_RT and NC_NRT boards, respectively, of the BLC chassis. These two windows allow the operator to interact with these two boards via the VxWorks shell. The window labeled "M333NRL" is a Unix "tip" window that connects to the M333NRL board on the BLC chassis. This window allows the operator to interact with the monitor program stored on the M333NRL board. The window labeled "SC" is for interacting with the operating system of the NFSE System console.

9.3.3.2 Interface Commands

The operator can issue VxWorks shell commands in the windows labeled "NC_RT" and "NC_NRT", Unix shell commands in the window labeled "SC", and MVME333Bug commands in the window labeled "M333NRL".

9.3.3.2.1 VxWorks Shell Commands

The VxWorks shell commands are described in the Wind River Systems documentation that describes the VxWorks system. The operator can obtain on-line help for a specific command by typing "vxman <command>".

9.3.3.2.2 Unix Commands

The Unix commands are described in the numerous books on this operating system. The operator can obtain on-line help for a specific command by typing "man <command>".

9.3.3.2.3 MVME333Bug Commands

The MVME333Bug commands are described in the manual that accompanies the MVME333Bug proms. The operator can obtain on-line help by typing "HE". The HE (help) command displays all commands available in MVME333Bug.

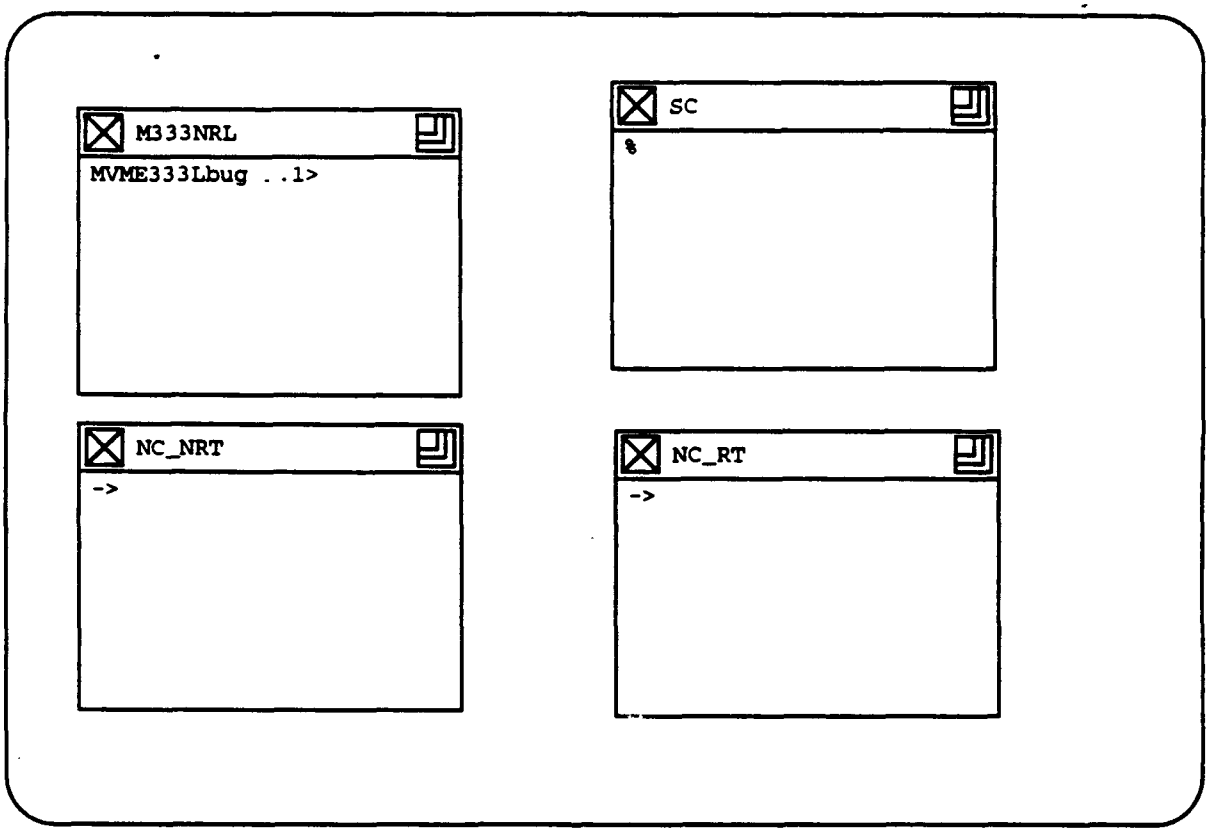


Figure 17 Startup display on System Controller screen.

9.3.4 System Controller/Link Controller Interface (IF_SC_LC)

The IF_SC_LC interface is an RS232 connection between Port A on the Sun console and Port 1 on the MVME705, which serves as a transition board for the M333NRL board.

9.3.5 System Controller/Network Controller Interface (IF_SC_NC)

The IF_SC_NC interface is an ethernet to which both the SC and NC are attached. The SC is connected directly to the ethernet, while the NC is connected to the ethernet via the LANCNTR board.

9.3.6 NC-to-NC Peer Interface (IF_NC_NC)

9.3.6.1 Interface Requirements

Network Controllers on different nodes communicate with each other via synchronous and asynchronous messages. Synchronous messages are sent at precisely known times. Asynchronous messages are subject to queuing delays, and therefore, the precise time that they will be transmitted is not predetermined.

9.3.6.2 Data Requirements

The formats of synchronous and asynchronous messages are given in section 11.3.4.

9.3.7 LC-to-LC Peer Interface (IF_LC_LC)

Link Controllers on different nodes communicate with each other via bursts of 6 blocks that contain M_BLKSIZE bytes of upper-layer data plus a 2-byte (first five blocks) or 1-byte (sixth block) block checksum. See references [9] and [10] for a discussion of the checksum procedure.

9.4 Quality Assurance

TBD

9.5 Preparation for Delivery

None.

9.6 Notes

None.

9.7 Revisions to Section 9

None.

10. SOFTWARE DESIGN

This section serves as the Software Design Document (SDD) for all of the CSCIs that make up the NFSE System.

10.1 Software Design for the Network Controller (NC) CSCI

10.1.1 Scope

10.1.1.1 Identification

Section 10.1 covers the software design for the Network Controller (NC) CSCI, which is a component of the Network Frequency Selection Experiment System.

10.1.1.2 System Overview

See section 5.1.2 on page 13.

10.1.1.3 Subsection Overview

Section 10.1 describes the design of the NC CSCI.

10.1.2 Referenced Documents

See Section 3.

10.1.3 Preliminary Design

10.1.3.1 NC Overview

10.1.3.1.1 NC Architecture

Figure 18 shows the NC architecture. The NC is organized as two components -- a non-real-time component NC_NRT and a real-time component NC_RT. Each of these components operates on a separate processor board. The NC_RT handles those tasks that are time critical. The remaining tasks are executed out of the NC_NRT processor. Within the NC are two major components -- the NFSS module and the NC_COM module. The NFSS implements the Network Frequency Selection Service, while the NC_COM module implements the communication, monitoring, and other services required of the NC. For the most part, the NC_COM design is identical to the NC design that was used by NRL in the UNT project. The major addition to the present NC design is the Network Frequency Selection Service (NFSS) functionality. A minor addition is the inclusion of the IP Connection function, which enables internetting between the NFSE's internal network and other networks. Figure 55 also shows the key interfaces internal to the NC.

Additional timing for the NC for operation with the NFSE System's internal network is given in figure 19. During the period between activations of the NFSS, time is divided into Reorganization Periods of duration T_{RP}. These are further divided into frames F_j, which in turn are subdivided into transmission

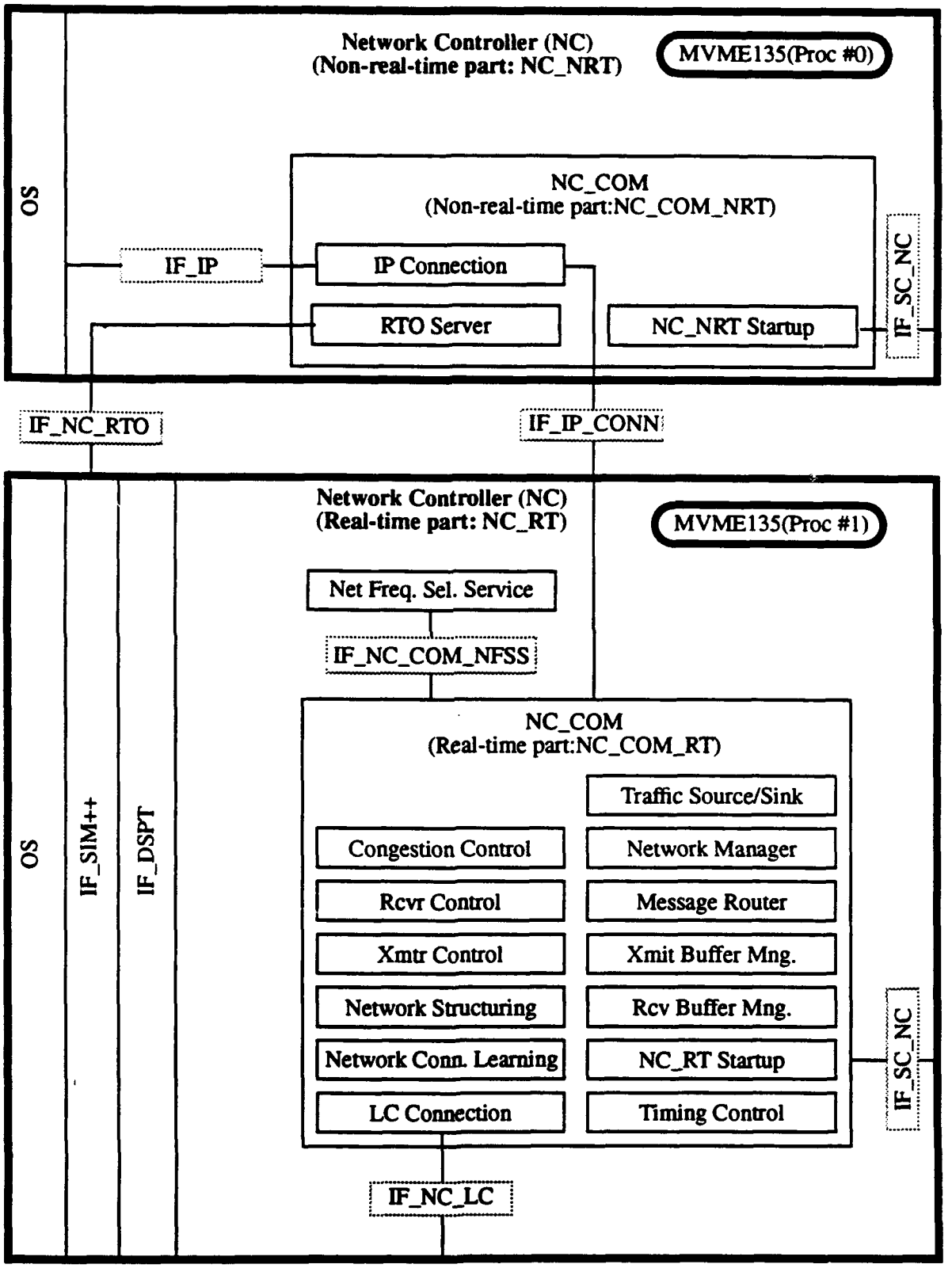


Figure 18 NC architecture showing CSCs, key interfaces, and the target hardware.

slots S'_j. In the event that a whole number of reorganization periods does not fit exactly into the interval between activations of the NFSS, a time-fill period T_{fill} has been added The number of frames per reorganization period is given by $N_{F'}$.

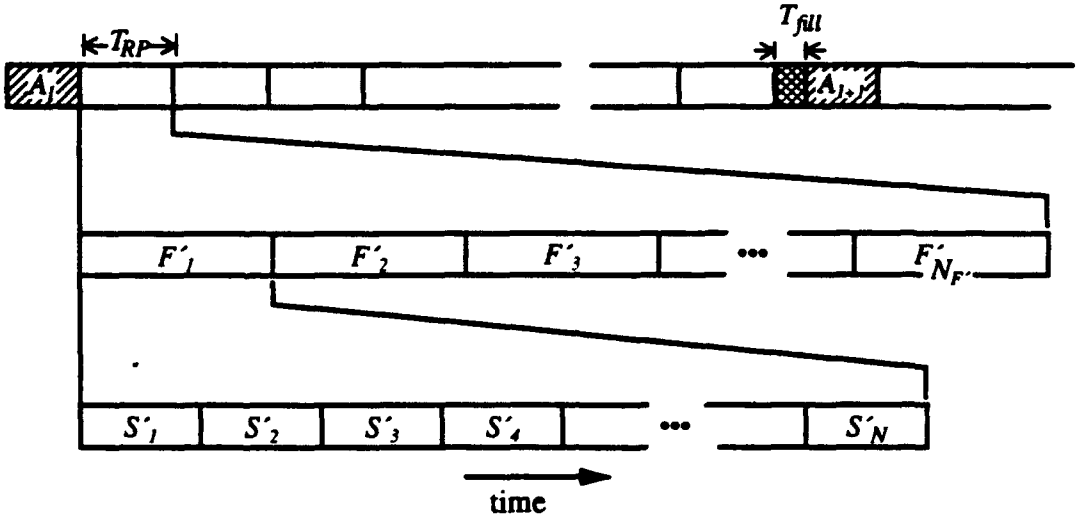


Figure 19 Additional system timing for operation with NFSE System's internal network.

10.1.3.2 NC Design Description

The NC CSCI is further decomposed into computer software components (CSC), which are described in this subsection. The computer software components of the NC are also shown figure 55.

10.1.3.2.1 Network Connectivity Learning (NC_NCL) CSC

The NC_NCL is used to determine the status of local connectivities, to identify "reliable" bidirectional links between the host node and its neighbors, and to disseminate link quality information to all nodes in the network.

The NC_NCL supports the network frequency selection service and the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.2 Network Structuring(NC_NS) CSC

The NC_NS is used to organize the nodes into a Linked Cluster Architecture, as described in [3].

The NC_NS supports the network frequency selection service and the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.3 Transmitter Control (NC_XC) CSC

This CSCI determines when a transmitter is allowed to transmit, and it provides an interface to the remote control functions of the transmitter.

The NC_XC supports the network frequency selection service and the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.4 Receiver Control (NC_RC) CSC

This CSCI provides an interface to the remote control functions of the receiver.

The NC_RC supports the network frequency selection service and the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.5 Transmit Buffers Management (NC_TBM) CSC

The NC_TBM has several functions - it manages the storage of messages awaiting transmission, it packs messages into transmit buffers, and it generates and inserts sequence numbers in relay message headers. The messages stored by the NC_TBM may originate in higher layers or they may originate locally in other NC CSCs. Also, it manages the storage of pointers to messages that are being relayed. Relay messages have their own message store facility based in the NC_RBM (Receive Buffers Management) CSC, however, the TBM accesses relay messages via queues of pointers that it maintains.

The NC_TBM supports the network frequency selection service and the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.6 Receive Buffers Management (NC_RBM) CSC

The Receive Buffers Manager checks each block of an incoming packet for block errors[1], computes a link quality index (LQI) value for each link, unpacks messages from each receiver's Receive Buffer and forwards each message to the appropriate destination within the NC. The NC_RBM, also, prevents duplicate messages from being delivered to the upper layer and removes relay messages from relay queues (i.e., from the NC_RBM message store) that have exceeded their time-to-live.

The NC_RBM supports the network frequency selection service and the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.7 Congestion Control (NC_CC) CSC

The function of Congestion Control is to prevent the degradation of network performance that can occur when traffic levels exceed that which the internal network can effectively handle.

The NC_CC CSC supports the network communication service requirements given in Section 5.3.2.1.

1. Checksums are computed by the Link Controller and the results sent to the Network Controller for use in determining message errors and link quality.

10.1.3.2.8 Message Router (NC_MR) CSC

The function of the Message Router (NC_MR) is to determine whether the host node should relay an asynchronous message and to forward asynchronous messages destined for the host node to the appropriate destination port.

The NC_MR supports the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.9 Network Manager (NC_NM) CSC

The Network Manager maintains the local performance database. NC_NM is responsible for monitoring system performance, and it is intended to serve as an interface for purposes of monitoring and controlling the operation of the internal network and other components of the NFSE System.

The NC_NM supports the network frequency selection service and the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.10 Network Frequency Selection Service (NC_NFSS) CSC

This CSC implements the network frequency selection service specified in section 5.3.2.1.1.1.

10.1.3.2.11 NC Timing Control (NC_TC) CSC

This CSC implements the timing structure shown in Figure 7. It also provides a facility for scheduling procedure calls and subsequently calling those procedures at the appropriate times.

The NC_TC supports the network frequency selection service and the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.12 NC_RT System Startup (NC_RT_SU) CSC

This CSC is the first NC software that is executed on the NC_RT processor board at system startup, and it is responsible for starting the rest of the NC_RT software.

The NC_RT_SU supports the network frequency selection service and the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.13 NC_NRT System Startup (NC_NRT_SU) CSC

This CSC is the first NC software that is executed on the NC_NRT processor board at system startup, and it is responsible for starting the rest of the NC_NRT software.

The NC_NRT_SU supports the network frequency selection service and the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.14 IP Connection (NC_IP_CONN) CSC

This CSC implements the connection to the Internet Protocol (IP) via the Operating System interface.

The NC_IP_CONN supports the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.15 LC Connection (NC_LC_CONN) CSC

This CSC implements the NC's part of the IF_NC_LC interface.

The NC_LC_CONN supports the network frequency selection service and the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.16 Real-time Output Server (NC_RTO_SV) CSC

This CSC allows concurrent file output capability in a multiprocessor, real-time system.

The NC_RTO_SV supports the network frequency selection service and the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.17 NC Sim++ Interface (IF_SIM) CSC

The IF_SIM CSC emulates a subset of the Sim++ software from Jade Simulations International (JSI) on a vxWorks target MVME135A CPU.

The IF_SIM CSC supports the network frequency selection service and the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.18 NC DSPT Interface (IF_DSPT) CSC

The IF_DSPT CSC emulates a subset of the DSPT software from NRL on a vxWorks target MVME135A CPU.

The IF_DSPT CSC supports the network frequency selection service and the network communication service requirements given in Section 5.3.2.1.

10.1.3.2.19 Traffic Source/Sink (NC_SRC_SNK) CSC

This CSC provides a source and sink of traffic in order to perform built-in tests of the NFSE System's internal network.

The NC_SRC_SNK supports the network communication service requirements given in Section 5.3.2.1.

10.1.4 Detailed Design

10.1.4.1 Network Connectivity Learning (NC_NCL) CSC

The NC_NCL CSC is structured as two component CSUs -- the Periodic Probing Algorithm (PPA) and the Network Connectivity Learning Algorithm (NCLA).

10.1.4.1.1 Periodic Probing Algorithm (PPA) CSU

10.1.4.1.1.1 Periodic Probing Algorithm Design Specification/Constraints

The purpose of the PPA is to provide a periodic check of a node's direct connectivities with other nodes in the network and to identify and report reliable bidirectional links to its neighbors.

10.1.4.1.1.2 Periodic Probing Algorithm Design

Each node can discover who its neighbors are by the process of probing. When a probe message is broadcast, every other node replies with either an acknowledgment (ACK) or a negative acknowledgment (NAK) depending on whether or not it received the probe message. A simple strategy for effecting such a probing by every node is to set up two TDMA frames for controlling the transmission of probe and acknowledgment messages. In this approach, nodes are numbered from 1 to N and each node is assigned its correspondingly numbered transmission slot, once in each frame. The slot assignments are included in the Communication Plan (COMPLAN), which is available to every node.

During frame 1 each node broadcasts its probe message, which includes the ACKs and NAKs of previously transmitted probe messages. During frame 2 each node ACKs or NAKs frame 1 probe messages sent since its own frame 1 transmission. The formats of these "synchronous" messages are shown in figures 59 and 60.

Each node maintains information about the status of all of its incoming links. This information relates to both the results of the periodic probing and to the number of 192-bit blocks (see section 11.3.3.1 for a discussion of blocks) received correctly over the link since the last probe. In subsequent sections we provide details of how this status is measured. Here it is only necessary to point out that link quality is specified in terms of a Link Quality Index (LQI), which can take on the values 0 (lowest quality) to 3 (highest quality). $LQI_{i,j}$ gives the quality of the link from node j to node i. A node announces the LQI values of its own incoming links as part of the Periodic Probing Algorithm.

The PPA also identifies reliable, bidirectional links at a node and broadcasts this information to neighboring nodes. The test that is applied at node i to ascertain whether it is bidirectionally connected to node j is slightly different depending on whether j is greater than or less than i.

If $i<j$, node i considers a link to node j to be a reliable, bidirectional (RB) link and considers node j to be a RB-neighbor (RBN) if:

$$LQI_{i,j} \geq LQI_r \qquad \text{(EQ 10)}$$

and

$$LQI_{j,i} \geq LQI_r \qquad \text{(EQ 11)}$$

and node i and node j received each other's probes.[2]

If $j<i$, node i considers a link to node j to be a reliable, bidirectional (RB) link and considers node j to be a RB-neighbor if:

$$LQI_{i,j} \geq LQI_r \quad \text{(EQ 12)}$$

and

$$LQI_{j,i} \geq LQI_r \quad \text{(EQ 13)}$$

and node i received node j's frame 2 transmission, which indicated that node j was RB-linked to node i.[3] The reason for using a slightly different criterion in this last step when $j<i$ is so that both i and j agree on whether the link between them is an RB-link. Above, LQI_r is that value of LQI that defines a reliable, unidirectional link. As part of its frame 2 transmission, each node transmits a vector identifying those nodes to which it has RB-links.

Pseudocode for the PPA is shown in Figure 20.

```
Use own link_quality_vector (from RBMA measurements)
Wait for one of the following events and process as indicated:
 Case start_of_reorganization_epoch:
  probes_heard=(0,...,0);
  link_quality_matrix=null matrix+own link_quality vector;
  reliable_2way_connectivity matrix=null matrix;
 Case transmit_in_frame1:
  Send probes_heard vector and own link_quality vector;
 Case received_frame1_transmission_from_node_j:
  probes_heard[j]=1;
  Store received link_quality vector into row j of link quality matrix;
  if (i<j and link i,j is reliable and bidirectional)
    reliable_2way_connectivity[i,j]=1;
 Case transmit_in_frame2:
  Send probes_heard and own reliable_2way_connectivity vectors;
 Case received_frame2_transmission_from_node_j:
  Store received reliable_2way_connectivity vector row j of matrix;
  if (j<i and link i,j is reliable and bidirectional)
    reliable_2way_connectivity[i,j]=1;
```

Figure 20 Pseudocode for the Periodic Probing Algorithm at node i.

2. When the network first starts up, only this last condition is tested.

3. When the network first starts up, only this last condition is tested.

10.1.4.1.2 Network Connectivity Learning Algorithm (NCLA) CSU

10.1.4.1.2.1 Network Connectivity Learning Algorithm Design Specification/Constraints

During frame 1 of the reorganization epoch, nodes report on the quality of reception over all incoming links since the last reorganization. The purpose of the NCLA is to relay this information to all nodes in the network.

10.1.4.1.2.2 Network Connectivity Learning Algorithm Design

The Periodic Probing Algorithm (PPA) is responsible for the first transmission of every link quality report. Both the NCLA and PPA read the link quality information sent in frame 1 by the PPA; thereafter, the NCLA handles subsequent retransmissions.

The link quality vectors could be sent as separate, broadcast messages, which the network delivers to all nodes. Such an approach would work, but it is inefficient to send lots of small, broadcast messages. Moreover, some of this information may be unchanged from epoch to epoch, and therefore it need not be retransmitted. The NCLA avoids these inefficiencies.

The format of the data portion of an asynchronous local message that contains link quality reports is shown in Figure 21. Several reports can be packed into one message. Each report is tagged by the most recent epoch number ($modulo\,(2^{NB\_NCLA\_EPOCH})$) for which the data applies. When a link quality report is received, the NCLA checks to see if it represents newer and different information than is currently in the node's database. Old or unchanged data is not retransmitted - with one exception, which we describe shortly. The NCLA also decides whether to retransmit data received over the air. The NCLA performs this determination in a way similar to that used by the Broadcast Traffic Routing Algorithm (BTRA) (see section 10.1.4.8.1.2) to decide whether to retransmit a regular broadcast traffic message.

Description of Field	bits	
a. Num. connectivity reports (NCR)	NB_NODE_ID	
b. ID of orig. reporting node	NB_NODE_ID	} NCR
c. Epoch ID	NB_NCLA_EPOCH	} NCR
d. LQI vector	NB_SINGLE_LQI*N	} NCR
••• Additional connectivity reports		

Data packet size (bits): NB_NODE_ID+NCR*(NB_NODE_ID+NB_NCLA_EPOCH+NB_SINGLE_LQI*N)

Figure 21 Format of NCLA data packet for a seven node (N=7) network.

An additional service performed by the NCLA is to retransmit link quality information, received in PPA transmissions from its immediate neighbors, which has not changed over long periods of time. The reason for doing this is to prevent the case of some nodes never learning full connectivity information owing to missing one of the NCLA transmissions of its neighbor. The period of time that must pass before

a report is retransmitted is determined by the maximum epoch number that can appear in a link quality report. When a node transmits a link quality vector, it saves a copy of the epoch number that accompanied this report. As newer, but otherwise identical reports are received as part of the PPA transmissions, potential relay nodes simply update the epoch number for which these link qualities apply. If the link quality vector remains unchanged while the epoch number goes full circle to the value it had when that report was last transmitted, the report will be marked as "not transmitted" and may be retransmitted. If the epoch number corresponding to a link quality vector in a node's NCLA database remains unchanged while the epoch number goes through a complete cycle, that link quality vector is archived, which means that it will never be retransmitted. Pseudocode for the NCLA is given in Figure 22 and Figure 23.

```
if report has been superseded then
  Ignore the one just received;
else if report is a duplicate Then
  if state != sent then
    Update nodes_that_heard_report;
    Update nodes_that_must_be_sent_to;
    If nodes_that_must_be_sent_to is empty AND state == RELAY Then
      state = SAVE;
else if report contains same connectivities but more recent epoch then
  if t is in probes_heard list
    Increment num_epochs_unchanged by 1;
    if num_epochs_unchanged==NCLA_MAX_EPOCHS_BETWEEN_RETRANSMISSIONS
      Clear nodes_that_heard_report;
      Set nodes_that_must_be_sent_to = EBB;
      State = RELAY;
  Update nodes_that_heard_report;
  Update nodes_that_report_must_be_sent_to;
  Update last_epoch_for_which_report_was_valid from report received;
  If nodes_that_must_be_sent_to is empty AND state == RELAY Then
    state = SAVE
else
  num_epochs_unchanged=0;
  Update link_qualities for reporting node;
  nodes_that_heard_report= RB_neighbors of transmitting node;
  nodes_that_report_must_be_sent_to =
    (EBB_neighbors-RB_neighbors of reporting node);
  Update last_epoch_for_which_report_was_valid from report received;
  If nodes_that_must_be_sent_to is empty Then
    state = SAVE
  else
    state = RELAY;
```

Figure 23 Pseudocode for Procedure handle_connectivity_report from node t.

```
//NCLA uses the following data structures:
  Use RB_neighbors lists from PPA;
  Use probes_heard list from PPA
  Use LQI vectors from PPA
  Use EBB_neighbors list from LCA;
  Integer current_epoch, range (0 to NCLA_MAX_EPOCH);
  Array (1:NUM_NODES) of connectivity_control_blocks,
  each block contains the following:
    State is one of (UNINITIALIZED,INITIALIZED,SAVE,RELAY,SENT,STALE);
    List of nodes_that_heard_report;
    List of nodes_that_report_must_be_sent_to;
    Integer epoch_in_which_report_was_relayed_by_this_node;
    Integer last_epoch_for_which_report_was_valid;
    Link qualities at reporting node;
Await occurrence of one of the following events and then process it:
  CASE event is start of epoch:
   Update current_epoch;
   FOR every report DO
   if state == SAVE OR state == RELAY OR state == SENT then
    if current_epoch == last_epoch_for_which_report_was_valid
     then state= STALE
  CASE event is transmission opportunity:
    Pack as many reports as possible into the NCLA message,
    that satisfy state == RELAY;
    For each report that is packed DO
      epoch_in_which_report_was_relayed_by_this_node=current_epoch;
      state= SENT;
  CASE event is NCLA message received from node t:
    Unpack connectivity reports and send them to handler;
  Case event is network reorganized:
    Get RB_neighbors and probes_heard lists from PPA;
    Get EBB_neighbors list from LCA;
    FOR all nodes in probes_heard list
      Create connectivity report from PPA database
      Call handle_connectivity_report;
    FOR every report DO
      if state == SAVE OR state == RELAY then
        Recompute nodes_that_report_must_be_sent_to;
        if nodes_that_report_must_be_sent_to list is empty then
          state= SAVE;
        else
          state= RELAY;
  Case event is connectivity report received:
    Call procedure handle_connectivity_report;
```

Figure 22 Pseudocode for the NCLA.

10.1.4.2 Network Structuring (NC_NS) CSC

This CSC is responsible for developing a network control structure.

10.1.4.2.1 Linked Cluster Algorithm (LCA) CSU

10.1.4.2.1.1 Linked Cluster Algorithm Design Specification/Constraints

The purpose of this algorithm is to organize the network into a Linked Cluster structure [3].

10.1.4.2.1.2 Linked Cluster Algorithm Design

The LCA allows each node to determine the role it should assume in the cluster structure and to learn who its "own" clusterhead is. In the version to be used in the NFSE System, gateway nodes are not selected. Instead, all nodes that are candidates to become gateway nodes become boundary nodes. Nodes that are neither clusterheads nor boundary nodes become terminator nodes. The LCA also provides each node with a list of RB-neighbors that are also members of the enhanced backbone (EBB) network - these are called EBB-neighbors. [3]

The LCA is piggybacked onto the Periodic Probing Algorithm in that it uses the same set of TDMA frames for communication and it relies on the bidirectional connectivity information gathered by the PPA.

Just before its frame 2 transmission, each node has obtained all the information it needs to implement the first step of the Linked Cluster Algorithm (that is, cluster formation). It is necessary, however, to include in its frame 2 transmission an announcement of the decision to become a clusterhead, if in fact the node has so decided. A node reveals this decision by including the ID of its own clusterhead in its frame 2 transmission. Each node selects as its own clusterhead the lowest numbered[4] clusterhead to which it is RB-connected. A node decides to become a clusterhead if it has not already heard from a clusterhead node to which it is RB-connected.

At the end of frame 2, nodes determine if they should assume the role of boundary nodes, and the EBB-neighbors list is derived. Pseudocode for the LCA is shown in figures 24, 25, and 26.

4. See the section on the PPA for a discussion of node numbering.

```
Use reliable_2way_connectivity matrix (see PPA);
Wait for one of the following events and process as indicated:
 Case start_of_reorganization_epoch:
  ownhead vector=(0,...,0);
  nodestatus=UNDEFINED;
  EBB_neighbors list=empty;
  heads_one_hop_away list=empty;
  heads_two_hops_away list=empty
 Case transmit_in_frame2:
  determine ownhead and store in ownhead[i];
  announce ownhead[i];
 Case received_frame2_transmission_from_node_j:
  ownhead[j]=ownhead reported by node j
  if (reliable_2way_connectivity[i,j]==1)
  { if (ownhead[j]==i)
      do nothing;
    else if (reliable_2way_connectivity[i,ownhead[j]]==1)
      put ownhead[j] into heads_one_hop_away;
    else
      put ownhead[j] into heads_two_hops_away;
  }
 Case end_of_frame2:
  if (node i is a CLUSTERHEAD)
    put any RB_neighbors of self into EBB_neighbors;
  linkup1();
  linkup2();
  if (nodestatus is not CLUSTERHEAD or BOUNDARY_NODE)
  { nodestatus=TERMINATOR_NODE;
    put ownhead[i] into EBB_neighbors;
  }
```

Figure 24 Pseudocode for the Linked Cluster Algorithm at Node i.

```
if (node i is not a CLUSTERHEAD)
{ //it is a candidate to be a boundary node
  if (there is at least one pair of nodes in heads_one_hop_away)
  { nodestatus=BOUNDARY_NODE;
    put nodes in heads_one_hop_away into EBB_neighbors;
  }
```

Figure 25 Pseudocode for procedure linkup1 at Node i.

```
if (node i is not a CLUSTERHEAD)
{ //node i is a candidate to become a boundary node;
  for all pairs of clusterheads (h1=ownhead[i],h2)
    where h2 is two hops from node i do
  { //check for prior linkage of h1 and h2;
    if (no RB_neighbor of node i is connected to both h2 and
      one of i's one_hop_away clusterheads)
    { nodestatus=BOUNDARY_NODE;
      put h1 into EBB_neighbors;
      copy all RB_neighbors whose ownhead is h2 into EBB_neighbors;
    }
  }
}
```

Figure 26 Pseudocode for procedure linkup2 at Node i.

10.1.4.3 Transmitter Control (XC) CSC

The Transmitter Control CSC is divided into two component CSUs - a Transmitter Control Driver and the Transmission Scheduling Algorithm.

10.1.4.3.1 Transmitter Control Driver (TCD) CSU

10.1.4.3.1.1 Transmitter Control Driver Design Specification/Constraints

The Transmitter Control Driver shall provide an interface for remotely setting the transmitter frequency.

10.1.4.3.1.2 Transmitter Control Driver Design

TBD

10.1.4.3.2 Transmission Scheduling Algorithm (TSA) CSU

10.1.4.3.2.1 Transmission Scheduling Algorithm Design Specification/Constraints

The Transmission Scheduling Algorithm shall cue the Transmit Buffer Management CSC when it is time to schedule transmissions by the Link Controller.

10.1.4.3.2.2 Transmission Scheduling Algorithm Design

A simple Fixed TDMA protocol is used to control transmissions in the NFSE System. The timing for this protocol is shown in figures 7 and 19. Each node is assigned a single slot during each transmission frame. Node *i* is always assigned Slot *i*.

10.1.4.4 Receiver Control (RC) CSC

The Receiver Control CSC contains one CSU - a Receiver Control Driver.

10.1.4.4.1 Receiver Control Driver (RCD) CSU

10.1.4.4.1.1 Receiver Control Driver Design Specification/Constraints

The Receiver Control Driver shall provide an interface for remotely setting the receiver frequency.

10.1.4.4.1.2 Receiver Control Driver Design

TBD

10.1.4.5 Transmit Buffers Management (TBM) CSC

The Transmit Buffers Management CSC contains one CSU - the Transmit Buffers Management Algorithm (TBMA) CSU.

10.1.4.5.1 Transmit Buffers Management Algorithm (TBMA) CSU

10.1.4.5.1.1 Transmit Buffers Management Algorithm Design Specification/Constraints

The TBMA has several functions - it manages the storage of messages awaiting transmission, it packs messages into transmit buffers, and it generates and inserts sequence numbers in relay message headers. The messages stored by the TBMA may originate in higher layers or they may originate locally in other NC algorithms. For example, the Network Connectivity Learning Algorithm generates messages that must be transmitted to other NCs. Also, it manages the storage of pointers to messages that are being relayed. Relay messages have their own message store facility based in the RBMA (Receive Buffers Management Algorithm), however, the TBMA accesses relay messages via queues of pointers that it maintains. The TBMA also handles the packing of messages into the transmit buffers for transmission during the next available transmission slot. The TBMA packs the buffers based on message lengths, types, and priorities. Finally, the TBMA generates and inserts message sequence numbers into the headers of all relay-type messages originating at the local NC. The reason to delay message sequencing to this point is because the TBMA determines the order in which messages are transmitted.

10.1.4.5.1.2 Transmit Buffers Management Algorithm Design

Each node has four transmit queues, the Send_local Queue, the Send_global Queue, the Broadcast_relay Queue, and the Point_to_point_relay Queue. The Send_local Queue contains messages originating at this node and being sent to nodes no farther than one hop away, i.e., no relaying required, and the Send_global Queue contains messages originating at this node and being sent to nodes that may be farther than one hop away, i.e., relaying may be required. The two remaining queues - Broadcast_relay and Point_to_point_relay - do not contain the actual relay messages. Rather, they contain message_tags that have the following attributes: 1) message_index (an index that locates the relay message in the RBMA's Message Store), 2) message_type, 3) priority, 4) length, and 5) time_in (the time the message was received by this

NC). In each queue, messages (or message_tags) are ordered by priority with the highest priority message at the head of the queue.

When a transmission slot opportunity arises, the Transmission Buffer Management Algorithm tries to fill the transmit packet buffer. The sequence of events that follows is described in the TBMA pseudo code (Figure 27). The fixed TDMA slots of Frames 1, 2, and 3 begin with synchronous data derived by NC algorithms (PPA, LCA, TSA, and CCA) with their remainders packed with asynchronous messages from the four transmit queues. All other slots contain only asynchronous messages. The first asynchronous message put into the packet is the highest priority message. If there is room for additional messages, the TBMA looks for the next highest priority message that will fit into the remainder of the packet and puts it into the slot. The process repeats until there are no more messages that can fit into the packet buffer. It is possible that, at times, the TBMA will not be allowed to accept traffic from the Send_global queue. The Congestion Control Algorithm may effect this blockage to prevent traffic congestion in the network or RBMA's limit on the number of messages that can be injected to prevent duplication of message sequence numbers may effect the blockage.

10.1.4.6 Receive Buffers Management (RBM) CSC

The Receive Buffers Management CSC contains one CSU - the Receive Buffers Management Algorithm (RBMA) CSU.

10.1.4.6.1 Receive Buffers Management Algorithm (RBMA) CSU

10.1.4.6.1.1 Receive Buffers Management Algorithm Design Specification/Constraints

The Receive Buffers Management Algorithm checks each block of an incoming packet for block errors[5], unpacks messages from each receiver's Receive Buffer, and forwards each message to the appropriate destination within the NC. The RBMA, also, prevents duplicate messages from being delivered to the upper layer and removes relay messages from relay queues (i.e., from the RBMA message store) that have exceeded their time-to-live.

10.1.4.6.1.2 Receive Buffers Management Algorithm Design

When a packet is received at the receive buffer, it has the same block format shown in figure 58 — minus the checksum bytes. It is unpacked block by block message by message, and it is checked for errors. Starting at the front of the receive buffer, the RBMA parses the receive buffer searching for messages. Any synchronous, control messages are located at the start of a buffer, and their lengths are known. The header of the first asynchronous message comes immediately after any synchronous messages, and therefore the location of this header is also known. The header contains information on the length of the message. Thus, the header of the first asynchronous message must be received correctly in order to determine where the next message header occurs. Thus, if the RBMA discovers an error in a block that contains a message header, all the remaining blocks in the packet must also be discarded. This is because each message header contains a message length indicator. Without the message length indicator it is impossible to tell where the current message ends and the following message in the packet begins. In fact, none of the remaining mes-

5. Error syndromes are computed by the Link Controller and sent to the Network Controller for use in determining message errors and link quality.

```
Compute pkt_pos // the position of the first available bit in the packet
                // where packing of asynchronous messages may begin
//Fill the remainder of the buffer with asynchronous messages
if cluster schedules are installed then
  while room for more messages do
    find the highest priority level max_pri of messages in the send_local,
      send_global, broadcast_relay, and point_to_point_relay queues;
    find the maximum length max_len of new traffic permitted;
    if RBMA limit on number of new messages is exceeded then
      max_len = 0;
    Request data from NTDS Port with priority > max_pri
      and length < max_len;
      Pack the NTDS message, if there is one;
      insert source_id and sequence_number into ntds_pkt header;
      compute pkt_pos, pkt_remainder, and traffic_limit;
    case send_global_q has msg and traffic_limit>=msg.length
      and pkt_remainder>=msg.length and msg.priority==max_pri:
      //Send a global msg (broadcast or pt2pt) sourced at this node
      insert source_id and sequence_number into msg header;
      remove msg from send_global_q;
      write msg into transmit buffer;
      compute pkt_pos, pkt_remainder and traffic_limit;
    case broadcast_relay_q has msg and
      broadcast_relay_q.priority >= pt2pt_relay_q.priority
      and broadcast_relay_q.priority >= send_local_q.priority
      and pkt_remainder >= msg.length:
      //Send a broadcast message relayed by this node
      remove msg from broadcast_relay_q;
      write msg into transmit buffer;
      call BTRA.msg_sent;
      compute pkt_pos, and pkt_remainder;
    case pt2pt_relay_q has msg and
      pt2pt_relay_q.priority >= send_local_q.priority
      and pkt_remainder >= msg.length:
      //Send a pt2pt message relayed by this node
      remove msg from pt2pt_relay_q;
      write msg into transmit buffer;
      compute pkt_pos, and pkt_remainder;
    case send_local_q has msg and pkt_remainder >= msg.length:
      //Send a local msg
      remove msg from send_local_q;
      write msg into transmit buffer;
      compute pkt_pos, and pkt_remainder;
    case all other cases:
      //Send a null message
      write null msg into transmit buffer;
      pkt_remainder = 0;
```

Figure 27 Pseudocode for the TBMA.

sages in the packet may be unpacked even if the remaining blocks are correct once synchronization with the message headers is lost.

When a packet's constituent messages are unpacked, they are routed internally to processes and queues within the NC based on the address given by the nc_port field in the message header. The pseudo code shown in Figure 28 and Figure 29 give the details of RBMA operation.

The synchronous data in the fixed-TDMA slots of frames 1, 2, and 3 are handled first, then the asynchronous messages are unpacked. End-of-data (i.e., Null) and Local messages cannot be relayed nor do they possess a Message Identifier (MID). Each point-to-point and broadcast message has a MID that is used for duplicate message detection and removal. The Message Identifier consists of the Source Address (address of the node that originated the message - NB_NODE_ID bits) and a Sequence Number - NB_SEQ bits.

The first step (see Figure 29) in processing the incoming stream of asynchronous message bits is to determine if more asynchronous messages can be unpacked. If fewer than two bits remain to be processed or if the next message is a Null message, then processing is finished because all messages have been processed. If messages remain to be processed, the length field of the next message is checked for errors. If it is error free, processing may continue; otherwise, processing must be terminated. If processing continues, the remainder of the message header and the message data field are checked for errors and, if all is well, the message is processed.

Local messages are merely routed to the internal NC process designated by the nc_port field of the message. However, if the message is a point-to-point or a broadcast message, RBMA must determine if the message is new or old. The algorithm makes this determination by using a MAX_NET_SIZE * MAX_-SEQ * 1 bit matrix and a queue that holds MSG_STORE_SZ MIDs. The bit matrix is initialized to zeros. When a message is received, its Source Address and Sequence Number are used as matrix indices to interrogate a single bit. If the bit is set to 0, the message is declared to be a new message, the bit is set to 1, and the message's MID is put into the queue. On the other hand, if the interrogated bit is set to a 1, the message is declared to be old. When an old MID pops out of the queue - which is a result of putting a new MID into a full queue - the MID is again used to select a bit in the matrix, only this time the bit is reset to 0.

New broadcast and point-to-point messages are put into the RBMA message store facility using a message store index computed by incrementing an index counter. Also, a message tag is created for each new message, which contains the index, the message type, message priority, message length, and a time stamp indicating when the message was received by the RBMA. The BTRA or P2PRA, depending on the message type, is then notified that the message has arrived. At the destination of a message, a copy of the message is delivered to the NC process designated by the NC port field.

The RBMA also implements a procedure to purge old messages in order to insure that a node never sees two different messages with the same MID at the same time. Our solution to this problem is to give each message a predetermined time-to-live at a relay node; call it t_l. Since the message tag, created when a message is unpacked from the receive buffer, contains the time that a message was received by RBMA, a background process running in RBMA can periodically cull the message store for messages exceeding their time-to-live at this node. If t_m is the time it takes to cycle through all of the sequence numbers at a node, then to prevent an old message from ever having the same sequence number as a new one, it is necessary that $t_m \geq t_l (N-1)$, where N is the number of nodes in the network. It is a function of the traffic throttling procedure of RBMA to insure that this condition is always satisfied, given N and t_l. Thus, the maximum average rate at which new messages can be generated is $n_s / (t_l (N-1))$ where n_s is the maximum sequence number.

```
case timing event for reading data from ftdma receive buffer:
  //RBMA pseudo code for unpacking the ftdma buffer
  update link quality information;
  if the ftdma receive buffer holds a packet then
    determine frame_num from CAA;
    determine sender from RSA;
    //unpack frames 1, 2, & 3 synchronous messages
    case frame_num = 1:
      unpack frame 1;
      call PPA, CAA, & CCA with sender & sync_msg;
    case frame_num = 2:
      unpack frame 2;
      call PPA, LCA, & CAA with sender & sync_msg;
    case frame_num = 3:
      unpack frame 3;
      call CAA with sender & sync_msg;
    unpack asynchronous messages;
case received packet from receiver:
  set holds_pkt to true;
```

Figure 28 Outer block pseudocode for RBMA

10.1.4.7 Congestion Control (CC) CSC

The Congestion Control CSC contains one CSU - the Congestion Control Algorithm (CCA) CSU.

10.1.4.7.1 Congestion Control Algorithm (CCA) CSU

10.1.4.7.1.1 Congestion Control Algorithm Design Specification/Constraints

The function of the Congestion Control Algorithm is to prevent the degradation of network performance that can occur when traffic levels exceed that which the network can effectively handle.

10.1.4.7.1.2 Congestion Control Algorithm Design

The Congestion Control Algorithm tries to maintain, globally, a maximum rate, g, at which new, network traffic can be transmitted by a node. *New, network traffic* is defined here to mean traffic that has not been previously transmitted and which the network is expected to route to its destination. Network traffic is distinguished from local traffic, which the network does not have to relay. This net-wide limitation on new, network traffic is shared by all nodes. That is, the limitation is based on a fictitious uniform loading of the network. From the computed rate g and the traffic loading at node i, node i is able to determine the maximum rate g_i at which it can transmit new network traffic. The rates g and g_i are measured in units of MAX_USER_DATA bits of information per FTDMA frame.

The interpretation of the traffic limit is as follows. Suppose the traffic limit g_i is 0.3. This means that every FTDMA schedule period node i can transmit, with probability 0.3, MAX_USER_DATA bits of new network traffic. Whether node i is allowed to transmit this additional new, network data, is determined by making a random draw from a uniform distribution over the range 0 to 1.0 at the beginning of every FTDMA frame. If the result of this draw is a number that is less than the fractional slot traffic limit (0.3 in this example), then the node is allowed to transmit MAX_USER_DATA bits of information during that frame.

```
//unpack asynchronous messages
while more messages in buffer do
  if less than 2 bits remain in buffer then
    more_messages is false
  else
    if message type field ok then
      get message type;
      if message type is NULL then
        more_messages is false
      else
      if message length field not ok then
        more_messages is false
      else
        get message length;
        if message header field ok then
          get nc port;
        if message data field ok then
          msg_ok is true;
        get reliability;
        if msg_ok or reliability is false then
          case message type is LOCAL:
            send message to nc port;
          case message type is BROADCAST:
            get message priority;
            get message source;
            get message sequence number;
            if message source <> this node then
              if message is not old then
                put message into message store;
                call BTRA.received_message;
                send message to nc port;
              else
                send call BTRA.received_message;
        case message type is PT2PT:
          get message priority;
          get message source;
          get message sequence number;
          get message destination;
          if message source <> this node then
            if message is not old then
              put message into message store;
              call P2PRA.received_message;
              if message destination == this node then
                send message to nc port;
```

Figure 29 Pseudo code for RBMA asynchronous message handling

The value of g is computed as follows. To allow varying the rate g, the Congestion Control Algorithm maintains two values for g, namely, a new value g_n and an old value g_o. At network startup g_n is initialized to CCA_INIT_GLOBAL_FACTOR. Thereafter, at the start of every CCA_EPOCHS_BETWEEN_INITS epochs, the Congestion Control Algorithm first sets $g_o = g_n$ and then resets g_n to CCA_MAX_GLOBAL_FACTOR. The CCA uses the following method to update g_n between these initializations. During frame 1 of each reorganization epoch each node i announces a measure, B_i, of the network traffic backlog at that node. Let the corresponding traffic load (in bits) be given by L. An estimate of the current value of L at a node is obtained from

$$L = \begin{cases} q_f - q_i + T, & \text{if } (q_f - q_i + T > 0) \\ \text{MAX\_USER\_DATA}, & \text{if } (q_f - q_i + T \le 0) \end{cases} \quad \text{(EQ 14)}$$

where q is the size of the network traffic backlog at the node, and the subscripts f and i refer to the final and initial values of q during the previous measurement epoch. T is the number of bits transmitted as network traffic during that same measurement period.

Let C stand for the available transmission capacity per epoch at the node, where each slot is presumed to represent MAX_USER_DATA bits of transmission capacity.[6] Thus, in the case of FTDMA, the available transmission capacity at each node is fixed at

$$C = \text{MAX\_USER\_DATA} * \text{NUM\_FRAMES\_PER\_EPOCH} \quad \text{(EQ 15)}$$

If L < C, that is, if there is excess capacity, there is the possibility that transmission queues at the node can be reduced and, under certain conditions, the net-wide, maximum allowed traffic loading on the network may be increased.The expected reduction in transmission queue size for the upcoming transmission epoch, q_c, is computed using (EQ 16).

$$q_c = \begin{cases} 0, & \text{if } (C - L < 0) \\ C - L, & \text{if } (0 \le C - L \le q_f) \\ q_f, & \text{if } (q_f < C - L) \end{cases} \quad \text{(EQ 16)}$$

Each node can compute a scale S, which is defined as follows.

$$S = (C - q_c)/L. \quad \text{(EQ 17)}$$

We define excess capacity x_c as follows.

$$x_c = \begin{cases} (C - L - q_c), & \text{if } (C - L - q_c > 0) \\ 0, & \text{if } (C - L - q_c \le 0) \end{cases} \quad \text{(EQ 18)}$$

Likewise, we define an excess scale S_x as

$$S_x = x_c/L \quad \text{(EQ 19)}$$

From the scales S and S_x each node computes, at the end of frame 1, a local value l for the maximum allowed traffic rate. At the outset of this computation l is set equal to g_o. One of three formulas is used depending on the values of S and S_x. If S is less than SWITCH, we use

---

6. Actually, each slot can hold slightly more than MAX_USER_DATA bits of information. A more accurate estimate of C would take into account the amount of synchronous message traffic sent in each slot.

$$l = g_0 (S/SWITCH)^2, \text{ if } S < SWITCH \quad \text{(EQ 20)}$$

otherwise

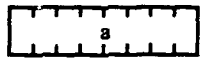
$$l = g_0 + RAMP (S_x - RAMP), \text{ if } RAMP < S_x < RAMP + 1 \quad \text{(EQ 21)}$$

or

$$l = g_0 + RAMP, \text{ if } S_x > RAMP + 1 \quad \text{(EQ 22)}$$

If $l < g_n$, the node sets the new global rate g_n to l.

Just prior to its own frame 2 transmission, each clusterhead does similar computations of l for itself and each of its RB-neighbors. It then sets l_{min} = minimum l for all of these nodes. If $l_{min} < g_n$ then the clusterhead sets $g_n = l_{min}$. Between initializations every CCA_EPOCHS_BETWEEN_INITS epochs the new value of g_n is sent to all nodes by the CCA using asynchronous, local messages. The format of these messages is shown in Figure 30. The method used to send these messages to all nodes is similar to that used to send broadcast messages using the Broadcast Traffic Routing Algorithm. When such a message is received by a node, the local g_n is updated to the value in the message - provided that value is less than the local value of g_n. Details of the protocol for processing incoming Congestion Control Messages are given in Figure 32.



Description of Field	bits
a. Global Congestion Factor	NB_GLB_CONGESTION_FACTOR
Data packet size (bits)	NB_GLB_CONGESTION_FACTOR

Figure 30 Format of a Congestion Control data packet containing an updated value for the global congestion control factor.

Non-uniform, new traffic loading is handled as follows. During frame 1 of each reorganization epoch, each node i announces an integer R_i, which is the amount of new, global traffic awaiting transmission. At the end of frame 2 each node computes a new maximum rate g_i for entering new traffic into the network.

$$g_i = (N' R_i g_o) / (\sum_j R_j) \quad \text{(EQ 23)}$$

where N' is 1 plus the number of reliable bidirectional neighbors of node i, and the summation extends over node i and all of its bidirectional neighbors.

Pseudocode for the Congestion Control Algorithm is given in Figure 31 and Figure 32.

```
//Data Structures used by CCA
Use RB_neighbors lists from PPA;
Use EBB_neighbors list from LCA;
float global_factor_step_size;
float go;
enum local_bc_msg_cntrl_blk_state
  {UNINITIALIZED, INITIALIZED, SAVE, RELAY, SENT, STALE};
struct cca_msg_control_block {
  int gn;
  List nodes_that_heard_report;
  List nodes_that_report_must_be_sent_to;
  local_bc_msg_cntrl_blk_state state;} cblk;
Await occurrence of one of the following events,
  then process as indicated below:
  case startup:
    cblk.gn=CAA_INIT_GLOBAL_FACTOR;
    cblk.nodes_that_report_must_be_sent_to=EMPTY;
    cblk.state=INITIALIZED;
    global_factor_step_size=
    CCA_MAX_GLOBAL_FACTOR/(2^NB_GLB_CONGESTION_FACTOR - 1)
  case start_of_an_initialization_epoch:
    go=cblk.gn*global_factor_step_size;
    cblk.gn=CAA_INIT_GLOBAL_FACTOR;
    cblk.state=INITIALIZED;
    tbma->set_traffic_limit(go);
  Case start_of_own_frame2_slot:
    if (s is a CLUSTERHEAD) {
      compute l for self and all RB neighbors;
      lmin=min of this set of l;}
    else {
      compute l for self;
      lmin=l;}
    if (lmin<cblk.gn)
     cblk.gn=lmin;
    cblk.nodes_that_heard_msg=EMPTY;
    cblk.nodes_that_report_must_be_sent_to=EBB_neighbors;
    cblk.state=RELAY;
 case received_cca_msg from node t
    handle_cca_msg(t,rcvd_gn);
  case network reorganized:
    Get RB_neighbors of neighbor nodes from PPA;
    Get EBB_neighbors list from LCA;
    if (cblk.state == SAVE OR cblk.state == RELAY) {
      Recompute olbk.nodes_that_report_must_be_sent_to;
      if (cblk.nodes_that_report_must_be_sent_to list is empty)
        cblk.state = SAVE;
      else
        cblk.state = RELAY;}
  case transmission_opportunity:
    if (cblk.state==RELAY) {
      send cblk.gn in a local message;
      cblk.state = SENT;}
```

Figure 31 Pseudocode for the CCA at node s.

```
procedure handle_cca_msg(int t, int rcvd_gn)
// t is the id of transmitting node
{
  if (rcvd_gngn > cblk.gn)
    ; //Ignore the cca message just received;
  else if rcvd_gngn == cblk.gn) {
    if (cblk.state!=sent) {
      Update cblk.nodes_that_heard_msg;
      Update cblk.nodes_that_must_be_sent_to;
      if (cblk.nodes_that_must_be_sent_to is empty AND
        (cblk.state == RELAY OR cblk.state ==INITIALIZED))
        cblk.state= SAVE;}
    else {//gn < cblk.gn
      Clear cblk.nodes_that_heard_msg;
    cblk.nodes_that_must_be_sent_to = EBB neighbors;
    cblk.state = RELAY;
    Update cblk.nodes_that_heard_msg;
    Update cblk.nodes_that_report_must_be_sent_to;
    If (cblk.nodes_that_must_be_sent_to is empty)
      cblk.state=SAVE}
}
```

Figure 32 Pseudocode for Procedure handle_cca_report.

10.1.4.8 Message Router (MR) CSC

10.1.4.8.1 Broadcast Traffic Routing Algorithm (BTRA) CSU

10.1.4.8.1.1 Broadcast Traffic Routing Algorithm Design Specification/Constraints

The function of the Broadcast Traffic Routing Algorithm is to send a broadcast message to all nodes in a network that is organized into Linked Clusters.

10.1.4.8.1.2 Broadcast Traffic Routing Algorithm Design

Each node follows the same rule to determine whether it should relay a broadcast message. The rule is to transmit the message if any EBB neighbors have not received the message already. But how do we determine when a neighbor has received a message? The HF Controller assumes that the RB neighbors of a transmitting node all receive that transmission. Consequently, for each broadcast message, a node maintains two lists. One list identifies to which nodes a message still needs to be sent - this list is initialized to a node's current EBB neighbors when the message is first received, and reinitialized after every reorganization. The second list identifies the nodes that are assumed to have already received the message - this list is updated every time the message is received. If there are any nodes on the first list when a transmission opportunity arises, then that message will be transmitted. Figure 33 contains pseudocode that describes the operation of the BTRA in more detail.

```
//BTRA uses the following data structures:
  Use RB_neighbors list from PPA - add self to this list;
  Use EBB_neighbors list from LCA;
  Array (0:MAX_SEQ) of Structure message_control_block
    List of nodes_that_heard_message;
    List of nodes_that_message_must_be_sent_to;
    Flag indicating whether message is_to_be_transmitted;
Await occurrence of one of the following events and then process it:
  Case event is network_reorganized:
    Get RB_neighbors list from PPA and add self to this list;
    Get EBB_neighbors list from LCA;
    For all none-empty message_control_blocks do
      Recompute nodes_that_message_must_be_sent_to;
      Update is_to_be_transmitted flag based on results of previous
        line;
      If is_to_be_transmitted flag changes then
        notify TBMA;
  Case event is broadcast_message_sent:
    Release message_control_block;
  Case event is message_deleted_from_local_archive:
    Release message_control_block;
  Case event is broadcast_message_received_over_the_air:
    If message is a duplicate Then
      If message_control_block is empty Then
        done processing event;
      Else when message_control_block is not empty Do
        Update nodes_that_heard_message;
        Update nodes_that_must_be_sent_to;
        If nodes_that_must_be_sent_to is empty Then
          If message is flagged to be sent Then
            Flag it not to be sent;
            Notify TBMA that message is not to be sent;
        Else when still need to send message to some nodes do
          If message is not flagged to be sent Then
            Flag it to be sent:
            Notify TBMA that message is to be sent;
    Else when message is not a duplicate do
      Create and initialize a new message_control_block;
      If nodes_that_must_be_sent_to is empty Then
        Flag message as not to be sent;
      Else when still need to send message to some nodes Do
        Flag message as to be sent;
        Notify TBMA to send message;
```

Figure 33 Pseudocode for the Broadcast Traffic Routing Algorithm.

10.1.4.8.2 Point-to-Point Routing Algorithm (P2PRA) CSU

10.1.4.8.2.1 Point-to-Point Routing Algorithm Design Specification/Constraints

The purpose of this algorithm is to compute the next relay node for point-to-point messages.

10.1.4.8.2.2 Point-to-Point Routing Algorithm Design

The point-to-point routing function is comprised of four important components. The first is the metric, which specifies the criterion for choosing a path. Next is the measurement process, which enables continuous monitoring of the network parameters needed for optimizing routes. The third is the updating protocol, which governs the dissemination of the measurements through the network. Last is the routing algorithm, which calculates the "best" route on the basis of the metric and the parameter measurements.In the sections covering the Transmit Buffer Management and Receive Buffer Management Algorithms we describe the measurement process used to determine the Link Quality Indexes, which are the measurements on which the P2PRA depends. In the sections covering the Periodic Probing and Network Connectivity Learning Algorithms we describe the protocols for disseminating these measurements to all nodes in the network. In this section we describe the routing metric and the routing algorithm.

It is typical for routing metrics to be based on throughput and delay. Minimum-hops routing is generally used in place of maximum throughput routing because it is much simpler to implement and approximates maximum throughput. However, minimum-hop routing treats links as either present or absent, when, in fact, link quality may vary over a wide range. Thus, for example, it may be preferable to route traffic over a highly reliable two-hop path instead of over an unreliable one-hop path. The Point-to-Point Routing Algorithm routes traffic based both on the number of hops required and path reliability. The P2PRA seeks paths that minimize the following metric.

$$1/(\prod h(LQI_{i,j}/LQI_{max})) \qquad \text{(EQ 24)}$$

The product is over all links along the path, and the hop factor h is a number less than 1. The hop factor is included to bias the metric in favor of paths with fewer hops. The Link Quality Indexes, provided by the PPA and NCLA, are used to determine path reliability.

The routing algorithm that is used is Dijkstra's shortest path algorithm.[10].

10.1.4.9 Network Manager (NM) CSC

10.1.4.9.1 Network Status Monitoring Algorithm (NSMA) CSU

The Network Status Monitoring Algorithm manages the local performance database. NSMA is responsible for archiving performance, and it is intended to serve as an external interface for purposes of monitoring and controlling the operation of the network.

10.1.4.9.1.1 Network Status Monitoring Algorithm Design Specification/Constraints

10.1.4.9.1.2 Network Status Monitoring Algorithm Design

Several CSUs of the Network Controller periodically send to the NSMA a status report. Each report contains data that characterizes the performance of that CSU. Some of this data may be archived for post analysis of network operation and some may be displayed at the System Operator's console. The following paragraphs summarize the contents of each of these reports. The paragraph header contains the name of the report and, in parentheses, when the report is sent to the NSMA.

- PPA Report (Sent in slot 1 of frame 3 of each epoch)
  RB_neighbors of this node;
  LQIs for all incoming links at this node;
  Frame 1 probes heard by this node;
- LCA Report (Sent in slot 1 of frame 3 of each epoch)
  Clusterheads one hop away;
  Clusterheads two hops away;
  EBB neighbors of this node;
  Node status (i.e., clusterhead, boundary, or terminator node);
  Own clusterhead's ID;
- CAA Report (Sent in slot 1 of frame 3 of each epoch) (UNT only)
  # number of cluster schedules in report;
  # cluster schedules in effect at this node;
  The following arrays range from 1 to CAA_MAX_NUM_CL_SCHEDS_REPORTED:
    array of schedule creator IDs;
    array of schedule sizes;
    array of transmission schedules;
    array of slot allocations;
- RSA Report (Sent in slot 1 of frame 1 of each epoch) (UNT only)
  Receiver utilization statistics
  # slots with insufficient number of receivers
- CCA Report (Sent in slot 1 of frame 3 of each epoch)
  Array of traffic limits requested;
  # bits of new-traffic sent;
  # bits new-traffic allowed;
- NCLA Report (Sent in slot 1 of frame 3 of each epoch)
  Link qualities vectors; (other nodes only)
  Vector of most recent update times for LQIs
- TBMA Report (Sent in slot 1 of frame 1 of each epoch)
  Statistics on messages sent;
  Statistics on message buffer utilization;
  Statistics on slot utilization;
- RBMA Report (Sent in slot 1 of frame 1 of each epoch)
  Statistics on messages received;
  Statistics on receive buffer utilization;

Statistics on block errors;
Log of block errors;
Log of garbled messages;

- BTRA Report (Sent in slot 1 of frame 1 of each epoch)
  Log of messages sent;
  Log of messages received
- P2PRA Report (Sent in slot 1 of frame 1 of each epoch)
  Log of messages sent;
  Log of messages received;
  Log of messages with no known path to destination;
  Routing metrics matrix;
  Array of next-hop relay nodes;
  Array of # hops on shortest paths;
- NDDS Report (Sent in slot 1 of frame 1 of each epoch)
  Log of Link-level performance;
- NFSS Report (Sent at end of each test cycle)
  Test Cycle Index Number
  For each frequency test cycle report the following:
  {
  Frequency Test Cycle Index Number
  Network's bidirectional connectivities
  Own Metric
  Test Frequency
  }
  Best Metric
  Best Frequency

10.1.4.10 Network Frequency Selection Service (NC_NFSS) CSC

The NC_NFSS CSC implements the Network Frequency Selection Service specified in section 5.3.2.1.1.1.

10.1.4.10.1 Network Frequency Selection Algorithm (NFSA) CSU

10.1.4.10.1.1 Network Frequency Selection Algorithm Design Specification/Constraints

The NFSA performs the following functions:

1. Selects frequencies to be tested during each test cycle of the NC_NFSS.
2. Sets transmitter and receiver to selected frequency at start of test of that frequency.
3. Disseminates connectivity information throughout the network.
4. Evaluates the frequency selection metric for each frequency tested.
5. Disseminates "best" frequency to other nodes in the network.
6. Sets transmitter and receiver to "best" frequency at end of test cycle.

10.1.4.10.1.2 Network Frequency Selection Algorithm Design

This subsection describes the design for implementing these functions.

10.1.4.10.1.2.1 Test Frequency Selection

The following relations are used to select the set of test frequencies for the *i*th activate cycle of the NFSS.

$$F_{sc} = \{f_j\}_{seq} \equiv \text{sequential collection of candidate frequencies} \quad \text{(EQ 25)}$$

where the *j*th element f_j can be accessed by the operator $at(j)$ That is,

$$f_j = F_{sc}.\text{at}(j) \quad \text{(EQ 26)}$$

Let M be the number of frequencies to be tested each cycle, and let $FA_i[]$ be an array of size M that holds the frequencies to be tested during the *i*th active cycle of the NFSS. We shall assume that the array bounds run from 0 to $M-1$. The last element of FA_i shall be filled as follows:

$$FA_i = \begin{cases} f_0 & , i = s \\ f_{b,i-1} & , i > s \end{cases} \quad \text{(EQ 27)}$$

where s corresponds o the index of the first active cycle of the NFSS at that node (see figure 3), and $f_{b,i-1}$ is the frequency selected a the "best" frequency during the previous active cycle of the NFSS. The first $M-1$ elements of FA_i shall be filled from F_{sc} as follows.

$$FA_i[k] = F_{sc}.at((i(M-1)+k)\ \%\text{sizeof}(F_{sc})),\ 0 \le m \le M-2 \quad \text{(EQ 28)}$$

where% is the modulo operator and sizeof (F_{sc}) is the function that returns the number of elements in the collection F_{sc}.

10.1.4.10.1.2.2 Set Transmit and Receive Test Frequencies

The NFSS calls routines in NC_COM to set the transmitter and receiver frequencies. Prior to testing with each frequency, there is a period of time reserved for switching transmitter and receiver frequencies. (See figure 7).

10.1.4.10.1.2.3 Disseminate Connectivity Information

The NFSA includes a variant of the protocol used by NCLA to disseminate connectivity information to all nodes in the network. One difference between the NFSA and NCLA protocols for disseminating connectivity information is the data that is passed. The NCLA passes 2-bit Link Quality Indexes among the nodes, while the NFSA passes bidirectional connectivity information. The latter is encoded as 1-bit per link. The NFSA obtains bidirectional connectivity information about itself and adjacent nodes from the Periodic Probing Algorithm (PPA); thereafter, the NFSA propagates this information to other nodes in the network.

Another, difference between the NFSA and NCLA protocols for disseminating connectivity information has to do with the way the connectivity database is initialized and the length of time that connectivity information, corresponding to a given frequency, is allowed to propagate throughout the network. In the NFSA, connectivity information corresponding to a given test frequency may only by disseminated while that frequency is being used. At the end of the period for testing a given frequency, the metric corresponding to that frequency is computed and the connectivity database is reinitialized.

The format of the data portion of an asynchronous local message that contains connectivity reports for the NFSA is similar to that used by the NCLA, which is shown in figure 21. One change is that the Epoch ID shown in the NCLA data packet is replaced by a Frequency Index field, as shown in figure 21. Another difference is that the NFSA uses only one bit per node to report connectivity information. Several reports can be packed into one message. Each report is tagged by a frequency index that identifies the frequency for which this report corresponds. When a connectivity report is received from the PPA, the NFSA checks to see if it represents different information than is currently in the node's database. Unchanged data is not retransmitted. The NFSA also decides whether to retransmit data received over the air in the form of NFSA messages. The NFSA performs this determination in a way identical to that used by the NCLA. Pseudocode for the NFSA protocol for disseminating connectivity information is given in figures 35 and 23.

	Description of Field	bits	
a.	Num. connectivity reports (NCR)	NB_NODE_ID	
b.	ID of orig. reporting node	NB_NODE_ID	} NCR
c.	Frequency Index	NB_FREQ_INDEX	
d.	Bidirectional Connectivities Vector	N	
•••	Additional connectivity reports		
	Data packet size (bits)	NB_NODE_ID+NCR*(NB_NODE_ID+NB_FREQ_INDEX+N)	

Figure 34 Format of NFSA data packet for a seven node (N=7) network.

```
//NFSA uses the following data structures:
  Use RB_neighbors lists from PPA;
  Use EBB_neighbors list from LCA;
  Integer current_freq_index;
  Array (1:NUM_NODES) of connectivity_control_blocks,
  each block contains the following:
    State is one of (UNINITIALIZED,SAVE,RELAY,SENT,STALE);
    List of nodes_that_heard_report;
    List of nodes_that_report_must_be_sent_to;
    Integer freq_index;
    RB_neighbors of reporting node;
Await occurrence of one of the following events and then process it:
  CASE event is start_of_frequency_test_cycle:
    Update current_freq_index;
    Initialize connectivity_control_blocks;
  CASE event is transmission_opportunity:
    Pack as many reports as possible into the NFSA message,
    that satisfy state == RELAY;
    For each report that is packed DO
      state= SENT;
  CASE event is NFSA_connectivity_message_received_from_node_t:
    Unpack connectivity reports for each
      Call handle_connectivity_report;
  Case event is network_reorganized:
    Get own RB_neighbors from PPA;
    Get EBB_neighbors list from LCA;
    FOR all RB_neighbor reports received from PPA database
      Call handle_connectivity_report;
```

Figure 35 Pseudocode for the NFSA protocol for connectivity dissemination.

```
if report.freq_index != current_freq_index then
  Ignore the report just received; // this is just a safety check
else if report is a duplicate of info in database Then
  if state != sent then
    Update nodes_that_heard_report;
    Update nodes_that_must_be_sent_to;
    If nodes_that_must_be_sent_to is empty AND state == RELAY Then
      state = SAVE;
else
  Update RB_neighbors for reporting node;
  nodes_that_heard_report= RB_neighbors of transmitting node;
  nodes_that_report_must_be_sent_to =
    (EBB_neighbors-RB_neighbors of reporting node);
  If nodes_that_must_be_sent_to is empty Then
    state = SAVE
  else
    state = RELAY;
```

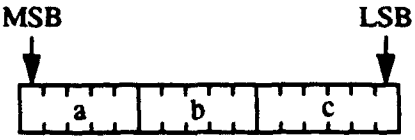
Figure 36 Pseudocode for NFSA procedure handle_connectivity_report from node t.

10.1.4.10.1.2.4 Frequency Selection Metric

There are several considerations that go into the selection of the "best" frequency. For example, is the network fully connected, that is, are there communication paths between all nodes? Are connectivities among certain nodes more important than connectivities among other nodes? How many relays are needed? Which frequency results in the highest throughput or the smallest message delays?

In this implementation of the NFSS we define the "best" frequency as the one that minimizes the frequency selection metric. Figure 37 defines frequency selection Metric A, which was the one initially formulated and tested (see section 14.4.1.1). Metric A is given in the form of an integer expressed in binary format. The most significant bit is on the left. The "best" frequency is the one that minimizes this metric. Thus, the most important factor in this metric is the number of unconnected nodes, n_{unc}, i.e., nodes that are disconnected from the largest component of the network. This implementation attempts to choose the frequency that leaves the fewest nodes disconnected from the largest component of the network. If several frequencies result in the same number of unconnected nodes, the best frequency is the one that requires the fewest number of relay nodes on the backbone network. This will maximize performance of a HAMA network. The least significant component in the frequency selection metric is the index of the frequency in the sequential collection of candidate frequencies F_{sc}.

After simulation tests, it became obvious that Metric A did not satisfactorily distinguish among network topologies that resulted in similar backbone networks. As a consequence Metric B was defined to correct this deficiency. Metric B is defined in the notes found in section 10.3.8.1.



Description of Field	bits
a. n_{unc}	NB_NFSS_UNCONN
b. Number of relay nodes on backbone network, N_{bb}	NB_NFSS_BB
c. Index of frequency in collection F_{sc}	NB_FREQ_INDEX

Figure 37 Format of frequency selection Metric A.

The pseudocode given in figure 38 can be used to find the connected components of the network. Typically, we would expect that the network would have only one component at the "best" frequency In the pseudocode of figure 38 we introduce two sets to hold intermediate results. $\{TN\}$ is the set of nodes of the component being built, and $\{TA\}$ is the set of nodes whose adjacencies have been considered by the algorithm. The components of the network are put into $\{C\}$. We define the function sizeofMaxComp as the function that returns the number of elements in the largest component of $\{C\}$.Thus, the equation

$$n_{unc} = N - sizeofMaxComp(\{C\}) \qquad \text{(EQ 29)}$$

provides the most significant field in the frequency selection metric.

```
Given: the undirected graph <V,E>
Find: the components, {C}, of <V,E>

{C} = empty set;
{W} = {1,2,...,N};
while ({W} is not empty)
{ {TN} = {TA} = empty set;
  transfer element el from {W} to {TN};
  while ({TN} != {TA})
  { select an element el2 from {TN} - {TA};
    transfer all neighbors of el2 from {W} to {TN};
    insert el2 in {TA};
  }
  insert copy of {TN} into {C};
}
```

Figure 38 Pseudocode to find the components of an undirected graph.

To find the next most significant field in the frequency selection metric, the algorithm given in figure 39 is used. This algorithm approximates the minimum backbone network (MBN) of the largest component of the network. The term N_{bb}, which appears in the frequency selection metric, is the number of nodes in this approximation of the minimum backbone network for the largest connected component.

```
find clusterheads, boundary nodes, and terminator nodes using the LCA
initialize MBN to empty set
while MBN is not connected to all nodes
  find non-terminator node that adds maximal neighbors to MBN
  add this node to MBN
weight each link (i,j) as follows:
  case (i,j) unconnected: weight INFINITY
  case i or j is TERMINATOR node: weight INFINITY
  case i and j on MBN weight 1
  case i on MBN or j on MBN weight 10
  otherwise weight 100
find shortest paths across network
use paths generated to connect MBN
add nodes used to connect to MBN
```

Figure 39 Pseudocode for finding the backbone network.

Finally, as a "tie-breaker", the least significant field in the selection metric is filled with the index of the frequency in the sequential collection F_{sc}.

10.1.4.10.1.2.5 Protocol for Disseminating "Best" Frequency Metric

The protocol for disseminating the metric for the "best" frequency is similar to that use by the Congestion Control Algorithm to disseminate the network traffic limit, g (see section 10.1.4.7.1.2). The pseudocode for this protocol is shown in figures 31 and 32. The messages containing the "best" frequency metric shall have precedence over the connectivity messages sent by the NFSA.

```
//Data Structures Used
Use RB_neighbors lists from PPA;
Use EBB_neighbors list from LCA;
unsigned int own_metric;
enum local_bc_msg_cntrl_blk_state
  {UNINITIALIZED,INITIALIZED,SAVE,RELAY,SENT,STALE};
struct nfss_metric_msg_control_block {
  unsigned int metric;
  List nodes_that_heard_metric;
  List nodes_that_metric_must_be_sent_to;
  local_bc_msg_cntrl_blk_state state;} cblk;
Await occurrence of one of the following events,
  then process as indicated below:
  case startup:
    cblk.metric = SHRT_MAX; //SHRT_MAX = 32767 = 0x7FFF
    cblk.nodes_that_heard_metric = EMPTY;
    cblk.nodes_that_metric_must_be_sent_to = EMPTY;
    cblk.state=INITIALIZED;
  case compute_metric:
    compute own_metric;
  case begin_flood_of_best_metric:
    cblk.metric=own_metric;
    cblk.state=RELAY;
  case received_metric_msg from node t
    handle_metric_msg(t,rcvd_metric);
  case network_reorganized:
    Get RB_neighbors of all nodes from PPA;
    Get EBB_neighbors list from LCA;
    cblk.nodes_that_metric_must_be_sent_to = EBB_neighbors;
    cblk.state = RELAY;
  case transmission_opportunity[1]:
    if (cblk.state == RELAY) {
      send cblk.metric in a local message;
      cblk.state = SENT;}
```

Figure 40 Pseudocode for the disseminating the "best" metric.

1. This corresponds to any transmission slot for the host node after the first frequency test cycle and exclusive of slots in frames 1 and 2.

10.1.4.10.1.2.6 Installation of Selected Frequency

The NFSS calls routines in NC_COM to set the transmitter and receiver to the frequency selected as the "best" frequency. Timing for this is shown in figure 7.

```
procedure handle_metric_msg(int t, unsigned short int rcvd_metric)
// t is the id of transmitting node
{
  if (rcvd_metric > cblk.metric)
    ; //Ignore the message just received;
  else if rcvd_metric == cblk.metric) {
    if (cblk.state!=sent) {
      Update cblk.nodes_that_heard_msg;
      Update cblk.nodes_that_must_be_sent_to;
      if (cblk.nodes_that_must_be_sent_to is empty AND
        (cblk.state == RELAY OR cblk.state ==INITIALIZED))
        cblk.state= SAVE;}
    else {//rcvd_metric < cblk.gn
      Clear cblk.nodes_that_heard_msg;
    cblk.nodes_that_must_be_sent_to = EBB neighbors;
    cblk.state = RELAY;
    Update cblk.nodes_that_heard_msg;
    Update cblk.nodes_that_report_must_be_sent_to;
    If (cblk.nodes_that_must_be_sent_to is empty)
      cblk.state=SAVE}
}
```

Figure 41 Pseudocode for Procedure handle_metric_msg.

10.1.4.11 NC_RT Timing Control (NC_TC) CSC

The NC_TC CSC implements timing control for the real-time component (NC_RT) of the Network Controller.

10.1.4.11.1 Timing Controller (TC) CSU

10.1.4.11.1.1 Timing Controller Design Specification/Constraints

The Timing Controller performs the following functions:

1. Provides for the scheduling and calling of time-scheduled procedure calls.
2. Provides other modules with timing information contained in system timing figures 3, 7, and 19.

10.1.4.11.1.2 Timing Controller Design

TBD.

10.1.4.12 NC_RT Startup (NC_RT_SU) CSC

The NC_RT_SU CSC implements those procedures necessary to get the real-time component (NC_RT) of the Network Controller in operation.

10.1.4.12.1 NC_RT Main (NC_RT_MAIN) CSU

10.1.4.12.1.1 NC_RT Main (NC_RT_MAIN) Design Specification/Constraints

The NC_RT_MAIN CSU performs the following functions:

1. Creates all NC_RT software objects.
2. Calls startup procedures in each object of NC_RT after all objects have been created.

10.1.4.12.1.2 NC_RT Main Design

TBD

10.1.4.13 NC_NRT Startup (NC_NRT_SU) CSC

The NC_RT_SU CSC implements those procedures necessary to get the non-real-time component (NC_NRT) of the Network Controller in operation.

10.1.4.13.1 NC_NRT Main (NC_NRT_MAIN) CSU

10.1.4.13.1.1 NC_NRT Main (NC_NRT_MAIN) Design Specification/Constraints

The NC_NRT_MAIN CSU performs the following functions:

1. Creates all NC_NRT software objects.
2. Calls startup procedures in each object of NC_NRT after all objects have been created.

10.1.4.13.1.2 NC_NRT Main Design

TBD

10.1.4.14 IP Connection (NC_IP_CONN) CSC

The NC_TC CSC implements timing control for the real-time component (NC_RT) of the Network Controller.

10.1.4.14.1 IP Driver (IP_DRIVER) CSU

10.1.4.14.1.1 IP Driver Design Specification/Constraints

The IP Driver shall provide an interface between the Internet Protocol, which is part of the host operating system, and the NFSE System's internal network.

10.1.4.14.1.2 IP Driver Design

The IP Driver design shall conform to the network interface driver design described in [12].

10.1.4.15 LC Connection (NC_LC_CONN) CSC

The NC_LC_CONN CSC implements that part of the IF_NC_LC interface that resides in the Network Controller.A single CSU, the Network Data Distribution System (NDDS), is used to implement the LC Connection.

10.1.4.15.1 Network Data Distribution System (NDDS) CSU

10.1.4.15.1.1 NDDS Design Specification/Constraints

The NDDS CSU design specification/constraints are given in [11]. A summary of these also appears in section 11.3.2.

10.1.4.16 Real-time Output Server (NC_RTO_SV) CSC

The NC_RTO_SV CSC handles the writing to file of data that has been placed in shared memory by the NC_SIM CSU file output facilities.

10.1.4.16.1 NC Output Server (NC_O_SV) CSU

10.1.4.16.1.1 NC Output Server Design Specification/Constraints

TBD

10.1.4.16.1.2 NC I/O Server Design

TBD

10.1.4.17 NC Sim++ Interface (IF_SIM) CSC

The IF_SIM CSC implements a portion of the Sim++ software from Jade Simulations International (JSI) on a vxWorks target MVME135A CPU.

10.1.4.17.1 NC Sim++ (NC_SIM) CSU

10.1.4.17.1.1 NC Sim++ Design Specification/Constraints

TBD

10.1.4.17.1.2 NC Sim++ Design

TBD

10.1.4.18 DSPT Interface (IF_DSPT) CSC

The IF_DSPT CSC implements a portion of the DSPT software from NRL on a vxWorks target MVME135A CPU.

10.1.4.18.1 NC DSPT (NC_DSPT) CSU

10.1.4.18.1.1 NC DSPT Design Specification/Constraints

TBD

10.1.4.18.1.2 NC_DSPT Design

This subsection describes the design for implementing these functions.

10.1.4.19 Traffic Source/Sink (NC_SRC_SNK) CSC

TBD

10.1.4.20 NC Timing Control (NC_TC) CSC

The NC_TC CSC implements timing control for the real-time component (NC_RT) of the Network Controller.

10.1.4.20.1 Timing Controller (TC) CSU

10.1.4.20.1.1 Timing Controller Design Specification/Constraints

TBD

10.1.4.20.1.2 Timing Controller Design

TBD

10.1.5 NC Data

The NC relies on data files for both input and output. This subsection identifies these files, indicates which CSC/CSUs use each file, and describe the file formats.

10.1.6 NC Data Files

The input files used by the NC are as follows:

complan.inp
timing.inp
nc_vx_start.inp

message.inp
flags.trace
nfss.inp

With the exception of complan.inp, the input files are the same at all nodes. The NC uses the same input files during simulation/emulation tests as it does in the field tests. For this reason, some of the parameters that are found in the input files have significance only when the NC is being tested in the simulator/emulator.

The NC uses the output files listed below.

log.out
stat.out
cksum_blk_rcvd.out
cksum_blk_good.out
msg.trace

10.1.6.1 Data File to CSC/CSU Cross Reference

10.1.6.2 Complan.inp

Figure 42 is an example of a typical complan.inp file. One item in this file that is unique to each node is the *node id*, on line 2. Figure 42 shows the complan.inp file for node 2.

```
1.1 Version of complan (do not change)
2 Node id
1 Send on ftdma only (true = 1, false = 0)
0 Send ntds using local msgs (true = 1, false = 0)
7 Number of nodes in network
0 0=start at 4-epoch boundary ; 1=start at time given below
0 0 0 0 (hr min sec ms) Time network is to start operating
01 01 1970 (mo day yr) date network is to start operating
01 01 1970 (mo day yr) today's date
10000 0 0 0 (hr min sec ms) Period network is to operate
1 Max. number of xmtrs that can be allocated this network
1 Number of xmtrs allocated this network
1 Set of xmtr ids allocated this network (integers)
1 Max. number of rcvrs that can be allocated this network
1 Number of rcvrs allocated this network
1 2 3 4 5 Rcvr ids
0 xmit code number (1 ...
40 xmtr power in watts
0 rcv code number (1 ....
0 AUTOMATIC_CONGESTION_CONTROL (1=ON, 0=OFF)
0.0 10 0.05 1.0 init traf lim(sl/fr),inc period(epochs),inc, max
13579 seed for traffic limit; //Added 9/8/89 by djb
0 use_icm 0=don't use 1=use
```

Figure 42 Sample complan.inp file from node 2.

10.1.6.3 Timing.inp

The timing.inp files contains values for most of the parameters that control network timing. Figure 43 shows a sample timing.inp file.

```
1.2 Version of timing.inp (do not change)
2400 xmtr info rate (e.g. 2400bps)
1056 xmtr packet length (data only - no cksum bits)
0                                xmtr interleaver depth
2400 rcvr info rate (e.g. 2400bps)
1056 rcvr packet length (data only - no cksum bits)
0 rcvr interleaver depth
4 TDM frames between epochs (integer, range > 3)
7 Number of slots per ftdma frame
6 num_cksum_blks_per_slot
650. t_ftdma_slot_duration (ms) 570 600
30. t_tune_rcvr (ms) relative to start of slot
-10. t_detune_rcvr (ms) relative to next t_tune_rcvr time
-200.0 T_PPA_INITIALIZATION //w.r.t. end of epoch
-195.0 T_LCA_INITIALIZATION //w.r.t. end of epoch
-190.0 T_TSA_INITIALIZATION //w.r.t. end of epoch
-185.0 T_RSA_INITIALIZATION //w.r.t. end of epoch
-180.0 T_PPA_LOAD_XMIT_BUFFER //w.r.t.
-175.0 T_LCA_LOAD_XMIT_BUFFER
-170.0 T_TSA_LOAD_XMIT_BUFFER //w.r.t.
-165.0 T_CCA_LOAD_XMIT_BUFFER //w.r.t. start of transmission slot
-90.0 T_TBMA_LOAD_XMIT_BUFFER //w.r.t. start of transmission slot
-80.0 T_SEND_PACKET (-80.0)
-0.4 T_INSTALL_CLUSTERS //w.r.t. end of frame 2
5.0 T_PPA_REPORT_UPDATE //w.r.t. end of frame 2
5.0 T_LCA_REPORT_UPDATE //w.r.t. end of frame 2
-10.0 T_CAA_REPORT_UPDATE //w.r.t. start of frame 4
5.0 T_NCLA_REPORT_UPDATE //w.r.t. end of frame 2
5.0 T_CCA_REPORT_UPDATE //w.r.t. end of frame 2
-7.0 T_RSA_REPORT_UPDATE //w.r.t. end of epoch
240.0 T_NSMA_REPORT_STATUS //w.r.t. end of epoch (-6.0)
10.0 T_ICM_POST_STATS //w.r.t start of current frame
350.0 T_READ_RCV_BUFFER //w.r.t. start of current slot (240.0)
-234.0 T_RCVR_TUNING //w.r.t.start of next slot - sched time for
635.0 T_RTS_DURATION (534.0) 547 575
0.0 T_RTS /*time to raise rts line */
```

Figure 43 Timing.inp file.

10.1.6.4 nc_vx_start.inp

The nc_vx_start.inp file, shown in Figure 49, contains parameters for setting the simulation duration (when the NC is being tested in simulation/emulation mode) and for setting the period between statistics reporting to the stat.out file. All times are given in milliseconds in nc_vx_startup.inp.

```
1500000.0 //simperiod in simulation time units
182000.0 //report update period in simulation time units
```

Figure 44 nc_vx_start.inp file.

10.1.6.5 Message.inp

The message.inp file contains canned messages for insertion into new, dummy traffic. The first line in this file contains text that is inserted into broadcast messages, and the second line contains text that is inserted into some local messages. Local messages are meant for nodes that are within direct range of the transmitting node, that is, they are messages that are not intended to be relayed by the network.

10.1.6.6 Flags.trace

Information in the file flags.trace is used to control what information is displayed on the console during tests and what gets archived. An example of flags set for a field test are shown in Figure 45.

```
 1 0 TRACE_EVP
 2 1 TRACE_LCA
 3 0 TRACE_CAA
 4 0 TRACE_RSA
 5 1 TRACE_CCA
 6 0 TRACE_NCLA
 7 1 TRACE_TBMA
 8 0 TRACE_RBMA
 9 0 TRACE_P2PRA
10 0 TRACE_BTRA
11 0 TRACE_NSMA
12 0 TRACE_NDDS
13 1 TRACE_PPA
14 0 TRACE_TC
15 0 TRACE_TC_NODE (1=trace_all_nodes, 0=trace_node_1)
16 0 TRACE_VALUES
17 0 TRACE_NTDSP
18 0 TRACE_IP
19 0 TRACE_MODEM
20 0 TRACE_ICM
21 1 LOG_PPA
22 1 LOG_LCA
23 1 LOG_CCA
24 1 LOG_NCLA
25 0 LOG_LINK_RCV_RATE
26 1 LOG_MSGS
29 0 TRACE_XMIT_FRAMES
30 0 TRACE_RCVD_FRAMES
31 1 TRACE_MSGS
32 0 TRACE_RCVR_USAGE
33 0 TRACE_CKSUM_BLK_ERRORS
34 0 TRACE_TIMINGS
35 0 TRACE_RCV_BLKS
36 0 TRACE_BLKS_NOT_RCVD
37 1 TRACE_LINK_RCV_RATE
999
```

Figure 45 flags.trace file used during tests.

10.1.6.7 Nfss.inp

The file nfss.inp is used to input the candidate frequencies to the NFSE System. Figure 46 illustrates the format for this file. Frequencies are entered in kilohertz, one frequency per line. The frequency list is terminated with a -1.

```
A         //Mode of operation: A, B, or C
0         //Use internal network: 0=No, 1=Yes
2862.0
4292.0
5312.5
5751.5
5915.0
6369.5
8566.0
8578.0
9250.0
-1
```

Figure 46 Sample nfss.inp file.

10.1.6.8 Log.out

Figure 47 shows a sample of a data written to the log.out file at node 2 every epoch, i.e., each time the network reorganizes. These traces contain information about the network connectivities and the network control structure. To fully understand the meaning of these outputs the reader should consult [3].

The first line shown in the example appears only if the NFSE System's internal network is operating and Congestion Control is turned on, and in that case it only appears every other epoch. The time of the status report is given as the millisecond of the day Greenwich Mean Time (GMT). For example, the time `78915440` shown in the sample report corresponds to 21h:55m:15.264s (GMT).

Next in the output file is a listing of the *reliable bidirectional neighbors* (RBN) of each node. Each row following the "RBN:" heading corresponds to a report of RBN for nodes 1 to 7, respectively. The connectivities are displayed as a hexadecimal representation of a 32-bit integer with leading zeros suppressed. To "read" the connectivities, it is first necessary to convert this hexadecimal number to the corresponding 32-bit binary equivalent. The most significant bit in the resultant binary number represents the existence or absence of a link to node 1; the second bit represents the existence or absence of a link to node 2, etc. Thus, for example, the hexadecimal value of `4000000` in row 1 (i.e., 0000 0100 0000 0000 0000 0000 0000 0000 binary) indicates that node 1 had a reliable bidirectional link only to node 6, since only the 6th bit is set to "1". (Note from the example that leading zeros in the 32-bit hexadecimal value are not printed.) If the reporting node is out of range, or if the reporting node has no RB-neighbors, a zero is printed in that row. A similar format is used to report on the probes heard during the last network reorganization; this information follows the RBN list. In this example, node 2 has heard probes from nodes 1, 3, 6, and 7. However, checking node 2's RBN row, we see that the link between nodes 1 and 2 is not a reliable, bidirectional one.

```
Installing traffic limit = 0.086274 at node 2
Status report for node 2 at time 78915440.000000
RBN:
  4000000
  26000000
  46000000
  0
  0
  e0000000
  68000000
Probesheard: a6000000

 LCA:
  nodestatus = 0
  hds1hop = 0
  hds2hops = 80000000
  EBBN = 26000000
  ownhead = 2
NCLA:
  LQI:
    300000
    c3c0000
    303c0000
    0
    c0000
    fc000000
    3cc00000
```

Figure 47 Sample of the data written after each reorganization epoch to the log.out file for node 2.

Next follows the report by the Linked Cluster Algorithm (LCA). The start of this report is indicated by the heading "LCA:". The LCA report gives a good indication of how the network is organizing itself locally. The *nodestatus* tells whether the reporting node is a *clusterhead* (0), *boundary node* (1), or a *terminator node* (2). At the time this sample report was made, node 2 had assumed the role of a clusterhead. Following the nodestatus are three lines of information giving: 1) clusterheads one-hop away, 2) clusterheads two-hops away, and 3) a list of nodes that are the reporting node's neighbors in the *enhanced backbone network* (EBBN). Nodes that are one-hop away can be reached directly, and nodes that are within two-hops can be reached via a single relay node. The formats of these three reports are the same as for the RBN. The neighbors on the EBBN are those nodes to which the reporting node will relay broadcast traffic. Thus, from the sample output, node 2 has no clusterheads within one hop (which is to be expected since node 2 is a clusterhead), and node 2 is aware that node 1 is a clusterhead that is two-hops away. Also, node 2 will seek to relay broadcast traffic to nodes 3, 6, and 7, as required. Finally, on the next line of output, the LCA reports the identity of its "own" clusterhead, which in this example is node 2 itself.

Next is the report from the Network Connectivity Learning Algorithm (NCLA). This report begins with the header "NCLA:". What follows is a report of the Link Quality Indexes (LQI) for each node in the network — one row per node. Thus, row 1 is a report of the quality of all incoming links at node 1; row 2 is a similar report for node 2, etc. Again the information is presented as a hexadecimal integer (32-bits)

with leading zeros suppressed. However, unlike the RBN information, two bits are used to represent the quality of a link. For example, the hexadecimal value `c3c0000` (i.e., 00 00 11 00 00 11 11 00 00 00 00 00 00 00 00 00 binary) in the second row would indicate that node 2 has LQI=3 (binary 11, the highest quality) on links from nodes 3, 6, and 7. Note that, unlike the RBN information, which is available only from neighboring nodes, the NCLA connectivity information is usually available for all nodes. Thus, for example, node 2 knows of node 5's connectivities even though it cannot hear node 5 directly.

The NFSA CSU also writes to the log.out file; it does so at the end of each test cycle. The format of this report is illustrated in figure 47. The first line identifies the reporting node and the time at which the report was written. The second line is a decimal number identifying the test cycle for which this report applies. For each frequency test cycle the following appears: frequency test cycle id (decimal), the bidirectional connectivities known by the node (RBN format), the metric (hexadecimal) computed at this node, and the test frequency (kilohertz). Following that the NFSA reports the "best" metric (hexadecimal) and the "best" frequency (kilohertz).

```
NFSS report for node 2 at time 78915440.000000
Test Cycle 875
Frequency Test Cycle 0:
Topology:
  4000000
  26000000
  46000000
  0
  0
  e0000000
  68000000
Own metric: 2081
Test Frequency: 4292.0
Frequency Test Cycle 1:
Topology:
  4000000
  26000000
  46000000
  0
  0
  e0000000
  68000000
Own metric: 2082
Test Frequency: 5312.5.0
Best Metric: 2081
Best Frequency: 4292.0
```

Figure 48 Sample of the data written by the NFSA CSU to the log.out file for node 2.

10.1.6.9 Stat.out

This file contains all the statistics collected during the run. Each line of output in the stat.out file contains the following: 1) time statistic was written to file, 2) type of statistic being reported, 3) platform, node, and object numbers of reporting object, 4) object type, 5) number of observations being reported, 6) average, maximum, minimum, and standard deviation for the statistic being reported.

10.1.6.10 Cksum_blk_rcvd.out

The Network Controller records in this file whether each block is received or not by the reporting node. A sample line from this output file, written at node 2, is shown in Figure 49. The first integer indexes the epoch being reported. In this example, the report is for the 357 *th* epoch recorded in this file. The next integer is the millisecond of the day (GMT) that this epoch started. Following that is information about which blocks were received by node 2 during the epoch. This information is output as an array of six hexadecimal 32-bit integers with leading zeros suppressed. Each bit represents one transmission block. A 1 means the block was received and a 0 means the block was not received. During each transmission slot a node transmits 6 blocks. There are 7 slots per frame and 4 frames per epoch for a total of 168 blocks per epoch. Thus, the last 6 zeros (representing 24 bits) on each line are to be neglected since they do not represent any connectivity information. The sample printout shows that node 2 received all blocks from nodes 1, 3, 5, 6, and 7 and no blocks from itself or node 4 during each of the four frames of this epoch

```
357 78915200.000000 FC0FC0FF FFFF03F0 3FFFFFC0 FC0FFFFF F03F03FF FF000000
```

Figure 49 Sample line of output from cksum_blk_rcvd.out at node 2.

10.1.6.11 Cksum_blk_good.out

The file cksum_blk_good.out is arranged in the same format as cksum_blk_rcvd.out. However, in cksum_blk_good.out we record the results of the checksum test on each received block. A 1 indicates the block was received with the correct checksum, and a 0 indicates that either the block was not received or it was received with an incorrect checksum. Together, cksum_blk_rcvd.out and cksum_blk_good.out characterize the overall behavior of the link layer of the Network. For the example shown in figures 49 and 49, all the blocks received from nodes 1 and 6 were received correctly. The following blocks were received with checksum errors: frame 1) blocks 1, 2, 4, and 5 from node 5 and block 1 from node 7, frames 2, 3, and 4) all blocks from node 5.

```
357 78915200.000000 FC0FC027 F7FF03F0 FFFFC0 FC003FFF F03F000F FF00000
```

Figure 50 Sample line of output from cksum_blk_good.out at node 2.

10.1.6.12 Msg.trace

The msg.trace file contains a line of output for each message transmitted or received by the Network Controller. More precisely, this file records those messages that were received by the Broadcast Traffic Routing Algorithm. Figure 49 is a small excerpt from a sample msg.trace file at node 1. Each line is formatted as follows: id of reporting node, software module id (see Figure 49), event type (see Figure 49), time event occurred, message type (see Figure 49), original creator of the message, and message sequence number. Thus, for example, the first line of output in figure 49 indicates that the Transmit Buffer Management Algorithm (TBMA) was responsible for transmitting a broadcast message (the 74 *th* message gener-

ated at node 1 since the last start-up) at time 80134510. The second line of output indicates that: the Broadcast Traffic Routing Algorithm (BTRA) at node 1 received a broadcast message for processing at time 80134700. This message was the 24 *th* message generated by node 5. Likewise, line 4 says that node 1 relayed this message at time 80139060.

Node id	Module id	Msg. event	time of event (ms of the day, GMT)	Msg. Type	Msg. Creator	Msg. Sequence #
1	60	0	80134510.000000	1	1	74
1	90	2	80134700.000000	1	5	24
1	90	2	80138600.000000	1	6	262
1	90	1	80139060.000000	1	5	24

Figure 51 Sample output from msg.trace at node 1.

```
#define NC_PPA 0
#define NC_LCA 10
#define NC_CAA 20
#define NC_RSA 30
#define NC_CCA 40
#define NC_NCLA 50
#define NC_TBMA 60
#define NC_RBMA 70
#define NC_P2PTRA 80
#define NC_BTRA 90
#define NC_NSMA 100
#define NC_NDDS 110
#define NC_NTDSP 120
```

Figure 52 C language definitions of software module ids. These appear in column 2 of the msg.trace file.

```
#define GEN_MSG 0
#define RELAY_MSG 1
#define RCV_MSG 2
```

Figure 53 C language definitions of message events. These event ids appear in column 3 of the msg.trace file.

```
#define BROADCAST_MSG 1
#define LOCAL_MSG 0
#define PT2PT_MSG 2
```

Figure 54 C language definitions of message types. These types appear in column 5 of the msg.trace file.

10.1.7 Requirements Traceability

TBD

10.1.8 Notes

None.

10.1.9 Revisions to Section 10.1

None.

10.2 Software Design for the Link Controller (LC) CSCI

The software design for the Link Controller is covered in references [7] to [10].

10.3 Software Design for the System Controller (SC) CSCI

10.3.1 Scope

10.3.1.1 Identification

Section 10.1 covers the software design for the System Controller (SC) CSCI, which is a component of the Network Frequency Selection Experiment (NFSE) system.

10.3.1.2 System Overview

See section 5.1.2 on page 13.

10.3.1.3 Subsection Overview

Section 10.1 describes the design of the SC CSCI.

10.3.2 Referenced Documents

See Section 3.

10.3.3 Preliminary Design

10.3.3.1 SC Overview

10.3.3.1.1 SC Architecture

Figure 55 shows the architecture of the System Controller. It consists of three CSCs: Archiver, Monitor, and SC Startup. It contains an RS232 interface for connecting to the Link Controller hardware (M333NRL) and an ethernet port for connecting to the two CPU boards used by the Network Controller, i.e., NC_RT and NC_NRT. The NFSE System operator has access to all of the features normally available to a user of a Sun Workstation. The System Controller includes a hard disk for storing archived data, NFSE System software, and NFSE System input files. The Monitor provides tools for monitoring and controlling the NFSE System. The operator uses tools that these CSCs and the Sun Unix environment provide in order to start, maintain, and stop the NFSE System.

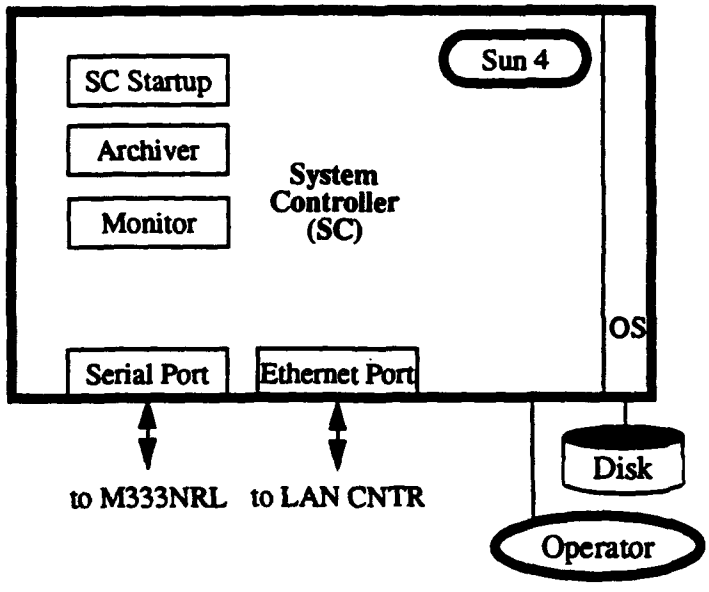


Figure 55 System Controller (SC) architecture showing CSCs, key interfaces, and the target hardware.

10.3.3.2 SC Design Description

The SC CSCI is decomposed into several computer software components (CSC), which are described in this subsection.

10.3.3.2.1 SC Startup (SC_SU) CSC

The SC Startup CSC is software used by the operator to start the NFSE System.

The SC_SU supports all of the requirements specified in section 5.3.2.1.

10.3.3.2.2 Archiver (SC_ARC) CSC

The Archiver CSC writes compressed copies of the input and output data files and names these files by concatenating the name of the host SC Sun and the time that the data was written.

The SC_ARC supports all of the requirements specified in section 5.3.2.1.

10.3.3.2.3 Monitor (SC_MON) CSC

The Monitor CSC provides a view of the state of the operating NFSE System. This is used to assist the operator in diagnosing system failures and to aid in assessing the overall "health" of the system.

The SC_MON supports all of the requirements specified in section 5.3.2.1.

10.3.4 Detailed Desig.1

10.3.4.1 SC Startup (SC_SU) CSC

The SC Startup CSC is made up of the NFSE Startup Scripts (NFSE_StartupScripts).

10.3.4.1.1 NFSE Startup Scripts (NFSE_StartupScripts)

10.3.4.1.1.1 NFSE Startup Scripts Design Specification/Constraints

The NFSE_StartupScripts perform the following functions:

- Provide windows for communicating to NC_RT, NC_NRT, and M333NRL, and SC hardware.
- Provide initializations for the NC_RT, NC_NRT, and M333NRL VME boards.

There are five startup scripts. The script for creating the four windows mentioned above is called SC_windows.script. Each of the four windows has a startup script associated with it. The names of these scripts are obtained by concatenating the window title and ".script". Thus, the four scripts are: SC.script, NC_RT.script, NC_NRT.script, and M333NRL.script. The following sections describe the functions provided by these startup scripts.

10.3.4.1.1.1.1 SC_windows.script

Figure 17 shows the four windows that are created. Their names identify the hardware to which these windows interface. Two of the windows are used to provide access to the MVME135A CPU boards used by the Network Controller. These boards are the NC_RT and NC_NRT, and they reside in the VME chassis of the Black Link Controller. The connections are accomplished by running the Unix rlogin program from the windows labeled NC_RT and NC_NRT. One window is connected to the console port of the MVME333 board named M333NRL, which also resides in the VME chassis of the BLC. This connection is obtained by running the Unix tip program from the M333NRL window. The fourth window, called SC, is used for running the Unix shell on the System Controller. The operator may open and close other win-

dows, as desired. This might be done to edit input files, examine output files, view source code, etc. There may be other windows on the screen also.

10.3.4.1.1.1.2 SC.script

The functions of SC.script are as follows:

- TBD

10.3.4.1.1.2 NC_RT.script

The functions of the NC_RT.script are as follows:

- TBD

10.3.4.1.1.3 NC_NRT.script

The functions of NC_NRT.script are as follows:

- TBD

10.3.4.1.1.4 M333NRL.script

The functions of M333NRL.script are as follows:

- TBD

10.3.4.2 Archiver (SC_ARC) CSC

10.3.4.2.1 Archiver Design Specification/Constraints

The Archiver shall provide software to facilitate the writing of any files ending in ".out", ".inp", or ".trace" to magnetic tape for later analysis.

10.3.4.2.2 Archiver Design

A unix shell script controls the archiving of all test results. This script is to be executed by each site's operator by typing the "save" command into the NC_NRT window at the end of each test run. All files in the current working directory ending in "inp", "trace", and "out" are compressed and written in unix tar format to file in the directory Tar.NFSE_results. The name given to the compressed file is formed by concatenating the name of the unix host at the site with the current date and time, e.g. athena.900927.072213 is the file archived at host "athena" on 9/27/90 at time (local host time) 07:22:13.

10.3.4.3 Monitor (SC_MON) CSC

10.3.4.4 Monitor Design

TBD

10.3.5 SC Data

TBD

10.3.6 SC Data Files

TBD

10.3.7 Requirements Traceability

TBD

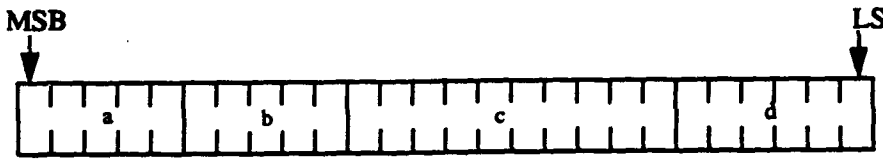
10.3.8 Notes

10.3.8.1 Refinements to the Frequency Selection Metric

During the NC CSCI evaluation (see section 14.4.1.1), the importance of refining the frequency selection metric to consider also the number of links in networks with similar backbones became apparent. The initial metric definition (Metric A) only takes into account the size of the required backbone. An improved metric (Metric B) introduces the number of links, as an additional measure of overall network connectivity to be considered as part of the metric evaluation.

10.3.8.1.1 Frequency Selection Metric B

Metric B adds the additional field "Number of Unconnected Links (NUL)" as the third most important consideration in selecting the best operating frequency. NUL is N*(N-1)/2 minus the known number of bidirectional links in the network. The expanded metric is shown in figure



Description of Field	bits
a. n_{unc}	NB_NFSS_UNCONN
b. Number of relay nodes on backbone network, N_{bb}	NB_NFSS_BB
c. NUL (Number of Unconnected Links).	NB_NFSS_UNCONN_LINKS
d. Index of frequency in collection F_{sc}	NB_FREQ_INDEX

Figure 56 Frequency selection Metric B.

10.3.9 Revisions to Section 10.3

None.

11. INTERFACE DESIGN DOCUMENT

11.1 Scope

11.1.1 Identification

This section serves a the Interface Design Document (IDD) for the NFSE system.

11.1.2 System Overview

See section 5.1.2 on page 13.

11.1.3 Section Overview

Section 11 specifies the detailed design for the interfaces between CSCIs of the NFSE system.

11.2 Referenced Documents

See Section 3.

11.3 Interface Design

11.3.1 Interface Diagrams

Figure 57 identifies the interfaces between the CSCIs of the NFSE System. Each of these interfaces is described in subsequent sections.

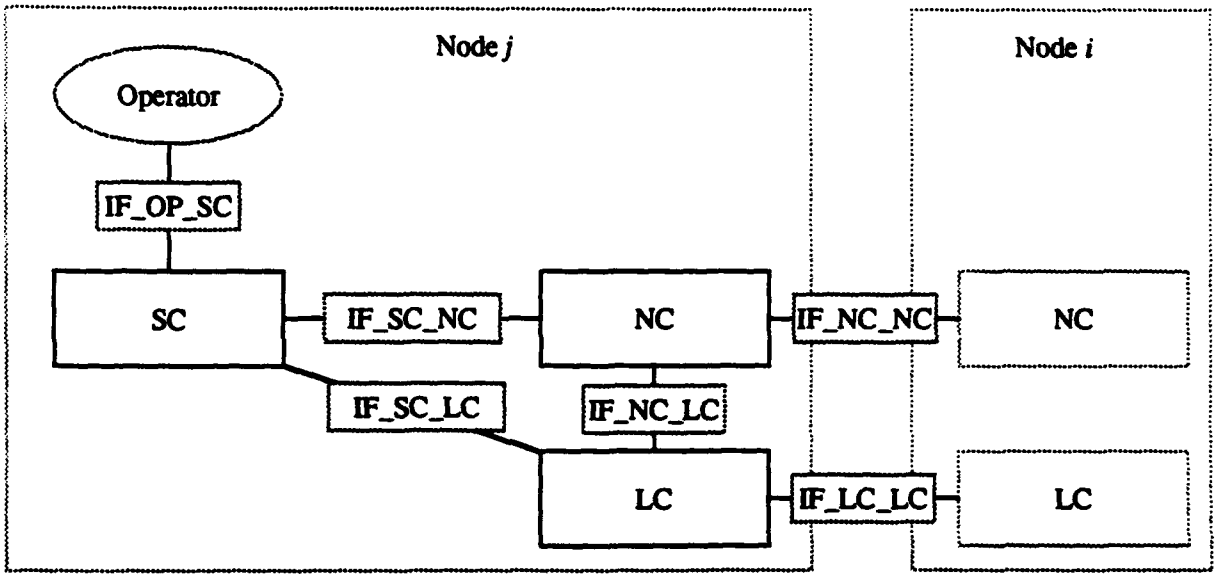


Figure 57 CSCI interfaces

11.3.2 NC/LC Interface (IF_NC_LC)

The following procedure calls are implemented by the NDDS and used by the NC to access LC resources. Also, the interface facilitates the delivery of data (i.e.. checksum blocks) and status information from the LC to the NC.

- PROCEDURE allocate_receive_string(nid,rid,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> rid: Integer id number of the receive string being allocated.
  - << error_code: Integer value returned where 0 means no error and >0 indicates id of NC to which rcvr is already allocated.
  - Called by NSMA at activation time.
  - Allocates a receive string to an NC.

- PROCEDURE deallocate_receive_string(nid,rid,error_code)
  - >>> nid: Integer id number of the requesting nc.
  - >>> rid: Integer id number of the receive string being deallocated.
  - << : Integer value returned where 0 means no error and >0 indicates id of another NC to which rcvr is allocated.
  - No calls are made to this NDDS procedure in the present NC implementation. It will have utility when multiple NC's share receive string resources.
  - Deallocates a receive string from an NC

- PROCEDURE allocate_transmit_string(nid,xid,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> xid: Integer id number of the transmit string being allocated.
  - << error_code: Integer value returned where 0 means no error and >0 indicates id of NC to which xmtr is already allocated.
  - Called by NSMA at activation time.
  - Allocates a transmit string to an NC

- PROCEDURE deallocate_transmit_string(nid,xid,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> xid: Integer id number of the transmit string being deallocated.
  - << error_code: Integer value returned where 0 means no error and >0 indicates id of another NC to which xmtr is allocated.
  - No calls are made to this NDDS procedure in the present NC implementation. It will have utility when multiple NC's share transmit string resources.
  - Deallocates a transmit string from an NC.

- PROCEDURE send_packet(nid,xid,packet,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> xid: Integer id number of the transmit string that the send_packet request is for.
  - >>> packet: Bitvector containing data for an entire packet.
  - << error_code: Integer error code indicator (0 -> no error, >0 -> error in NC/NDDS, <0 -> error in LC)
  - Called by TBMA when a transmission packet is ready to send.
  - Hands an entire packet to a transmission string.

- PROCEDURE prepare_to_send_next_slot(nid,xid,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> xid: Integer id number of the transmit string.
  - << error_code: Integer error code indicator (0 -> no error, >0 -> error in NC/NDDS, <0 -> error in LC)
  - Called by ES (scheduled by CAA) prior (~200 ms) to the next transmission slot.
  - Informs the LC Timing Controller to raise the modem's RTS line high at the next RTS time.
- PROCEDURE set_slot_timing(nid,xid,t_first_rts,slot_duration,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> xid: Integer id number of the transmit string.
  - >>> t_first_rts: Real number giving the second of the day when the first RTS event should occur.
  - >>> slot_duration: Real number giving the duration of a transmission slot in seconds.
  - << error_code: Integer error code indicator (0 -> no error, >0 -> error in NC/NDDS, <0 -> error in LC)
  - Called by NSMA at initialization time.
  - At initialization time, gives the LC Timing Controller information required for the computation of RTS and rcvr channel switching event times.
- PROCEDURE set_xmtr_packet_length(nid,xid,length,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> xid: Integer id number of the transmit string.
  - >>> length: Integer specifying the transmission packet length in bits (without checksum bits added).
  - << error_code: Integer error code indicator (0 -> no error, >0 -> error in NC/NDDS, <0 -> error in LC)
  - Called by NSMA at initialization time.
  - At initialization time, gives packet length information to the Xmit Error Detection Algorithm.
- PROCEDURE set_xmtr_nb_channel(nid,xid,chan_num,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> xid: Integer id number of the transmit string.
  - >>> chan_num: Integer code identifying a narrowband transmission channel.
  - << error_code: Integer error code indicator (0 -> no error, >0 -> error in NC/NDDS, <0 -> error in LC)
  - Called by NSMA at initialization time.
  - At initialization time, tells LC which channel to switch the transmitter to.
- PROCEDURE set_xmtr_inforate(nid,xid,bps,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> xid: Integer id number of the transmit string.
  - >>> bps: Integer giving the transmission rate in bits per second.
  - << error_code: Integer error code indicator (0 -> no error, >0 -> error in NC/NDDS, <0 -> error in LC)
  - Called by NSMA at initialization time.
  - At initialization time, tells LC what transmission rate to set for the transmit side of the modem

designated by XID.

- PROCEDURE set_xmtr_interleaver(nid,xid,num_frames,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> xid: Integer id number of the transmit string.
  - >>> num_frames: Integer specifying the number of 20ms modem frames for the interleaver depth.
  - << error_code: Integer error code indicator (0 -> no error, >0 -> error in NC/NDDS, <0 -> error in LC)
  - Called by NSMA at initialization time.
  - At initialization time, tells LC how to set the modem's transmit side interleaver depth.
- PROCEDURE set_xmtr_power(nid,xid,watts,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> xid: Integer id number of the transmit string.
  - >>> watts: Integer specifying the transmitter power level in watts.
  - << error_code: Integer error code indicator (0 -> no error, >0 -> error in NC/NDDS, <0 -> error in LC)
  - At initialization time, tells LC how to set xmtr power level.
- PROCEDURE set_rcvr_packet_length(nid,rid,length,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> rid: Integer id number of the receive string.
  - >>> length: Integer specifying the transmission packet length in bits (without checksum bits added).
  - << error_code: Integer error code indicator (0 -> no error, >0 -> error in NC/NDDS, <0 -> error in LC)
  - Called by NSMA at initialization time.
  - At initialization time, gives packet length information to the Rcv Error Detection Algorithm and the modem's receive side.
- PROCEDURE set_rcvr_nb_channel(nid,rid,chan_num,error_code)
  - >>> nid: Integer id number of the requesting NC
  - >>> rid: Integer id number of the receiver being allocated.
  - >>> chan_num: Integer code identifying a narrowband transmission channel.
  - << error_code: Integer error code indicator (0 -> no error, >0 -> error in NC/NDDS, <0 -> error in LC)
  - Called by RSA prior (~ 300 ms) to the slot for which the receiver must be retuned.
  - Tells LC Timing Controller which narrowband channel to set the receive to at the next receiver tuning time.
- PROCEDURE set_rcvr_inforate(nid,rid,bps,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> rid: Integer id number of the receiver being allocated.
  - >>> bps: Integer giving the transmission rate in bits per second.
  - << error_code: Integer error code indicator (0 -> no error, >0 -> error in NC/NDDS, <0 -> error in LC)

- Called by NSMA at initialization time.
- At initialization time, tells LC what transmission rate to set for the receive side of the modem designated by RID.

- PROCEDURE set_rcvr_interleaver(nid,rid,num_frames,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> rid: Integer id number of the receiver being allocated.
  - >>> num_frames: Integer specifying the number of 20ms modem frames for the interleaver depth.
  - << error_code: Integer error code indicator (0 -> no error, >0 -> error in NC/NDDS, <0 -> error in LC)
  - Called by NSMA at initialization time.
  - At initialization time, tells LC how to set the modem's receive side interleaver depth.
- PROCEDURE read_signal_quality(nid,rid,quality,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> rid: Integer id number of the receiver being allocated.
  - << quality: Integer value that quantifies signal quality.
  - << error_code: Integer error code indicator (0 -> no error, >0 -> error in NC/NDDS, <0 -> error in LC)
  - Not called in the present implementation. NSMA may use this in a later implementation.
  - Respond to a request for modem generated signal quality information.
- PROCEDURE read_rcvr_nb_channel(nid,rid,chan_num,error_code)
  - >>> nid: Integer id number of the requesting NC.
  - >>> rid: Integer id number of the receiver being allocated.
  - << chan_num: Integer code identifying a narrowband transmission channel.
  - << error_code: Integer error code indicator (0 -> no error, >0 -> error in NC/NDDS, <0 -> error in LC)
  - Called by RSA.
  - Return the channel setting of the receiver identified by RID.

11.3.3 LC/LC Interface (IF_LC_LC)

Information is exchanged between Link Controllers in packets that fit into one transmission slot. The format of this packet is described below.

11.3.3.1 Link-Level Packet (LC_PKT)

A link-level transmission (LC_PKT) is organized into 192-bit blocks, as shown in figure 58. Each block consists of 176 "data" bits followed by a 16-bit checksum (except for the last block which has only an 8-bit checksum)[1]. The number of blocks is determined by the duration of a transmission slot and by the information rate of the transmitter.

1. The smaller checksum on the last block was necessitated by implementation details relating to synchronizing to the modem used in the UNT demonstration.

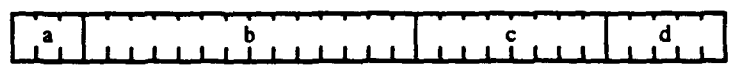
Figure 58 Structure of link-level transmission.

11.3.4 NC/NC Interface (IF_NC_NC)

11.3.4.1 Synchronous Messages

11.3.4.1.1 Frame 1 Synchronous Message (SMSG_F1)

Figure 59 shows the format of synchronous messages sent out during frame 1. Note that the first two message fields vary in length depending on the ID number of the transmitting node and the number of nodes in the network. The formats shown in Figure 59 and several of the following figures are for a seven-node network and assume node i (i = 4) is transmitting. If the description of particular message field is followed by "(UNT)", this indicates that these bits are not used by the NFSE System.

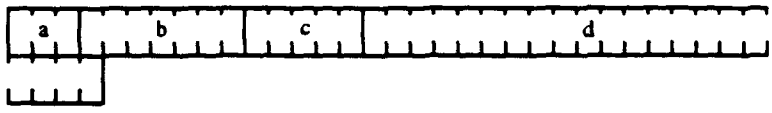


Description of Field	bits
a. Frame 1 ACK/NAK for PPA	i-1
b. LQI vector for node i	NB_SINGLE_LQI*N
c. Num. slots requested (UNT)	NB_SLOTS_REQUESTED
d. New traffic lim. (requested)	NB_TRAFFIC_LIMIT_REQUESTED
Msg. size (bits)	NB_SINGLE_LQI*N+i-1+NB_SLOTS_REQUESTED +NB_TRAFFIC_LIMIT_REQUESTED

Figure 59 Control message for frame 1 transmission from Node i (i=4, N=7)

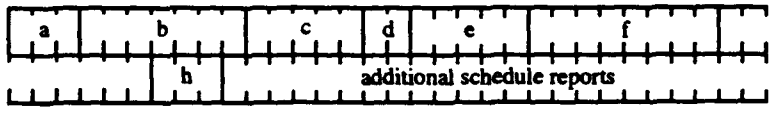
11.3.4.1.2 Frame 2 Synchronous Messages (SMSG_F2)

The format of the transmission buffer for the frame 2 transmission differs depending on whether the transmitting node is a clusterhead or a non-clusterhead. The simpler format is the one used by clusterheads; it is shown in Figure 60. Figure 61 shows the corresponding format for a non-clusterhead node.



Description of Field	bits
a. Fame 2 ACK/NAK for PPA	N-i
b. RB-neighbors	N
c. ID of own clusterhead	NB_NODE_ID
d. Cluster schedule (UNT)	NB_SLOTS_ALLOCATED*N
Msg. size (bits)	2N+NB_SLOTS_ALLOCATED*N-i+NB_NODE_ID

Figure 60 Control message for frame 2 transmission from clusterhead Node i (i=4, N=7)



Description of Field	bits	
a. Frame 2 ACK/NAK for PPA	N-i	
b. RB-neighbors	N	
c. ID of own clusterhead	NB_NODE_ID	
d. Num. schedules reported (R) (UNT)	NB_NUM_SCHEDS_REPORTED	
e. Schedule ID (UNT)	NB_NODE_ID	} R
f. Schedule length (UNT)	NB_NODE_ID+NB_SLOTS_ALLOCATED	
g. Pos. in schedule (UNT)	NB_NODE_ID+NB_SLOTS_ALLOCATED	
h. Slots received (UNT)	NB_SLOTS_ALLOCATED	
Msg. size (bits)	2N-i+NB_NODE_ID+NB_NUM_SCHEDS_REPORTED +3R(NB_NODE_ID+NB_SLOTS_ALLOCATED)	

Figure 61 Control message for frame 2 transmission from non-clusterhead Node i (i=4, N=7)

11.3.4.1.3 Frame 3 Synchronous Messages (SMSG_F3_CH) and (SMSG_F3_NON_CH)

As was the case for frame 3, the packet formats for frame 3 differ for clusterheads and non-clusterheads. Figure 62 shows the format used by the former, and Figure 63 shows the format used by the latter.

Description of Field	bits
a. Node status 0 = Clusterhead 1 = Terminator Node 2 = Boundary Node 3 = Gateway	NB_STATUS
Msg. size (bits)	NB_STATUS

Figure 62 Control Message for frame 3 transmission from a clusterhead.

Description of Field	bits	
a. Node status	NB_STATUS	
b. Num. schedules rpt. (R) (UNT)	NB_NUM_SCHEDS_REPORTED	
c. Schedule ID (UNT)	NB_NODE_ID	} R
d. Schedule length (UNT)	NB_NODE_ID+NB_SLOTS_ALLOCATED	} R
e. Pos. in schedule (UNT)	NB_NODE_ID+NB_SLOTS_ALLOCATED	} R
f. Slots received (UNT)	NB_SLOTS_ALLOCATED	} R
••• Additional schedules		
Msg. size (bits)	NB_STATUS+NB_NUM_SCHEDS_REPORTED +3R(NB_NODE_ID+NB_SLOTS_ALLOCATED)	

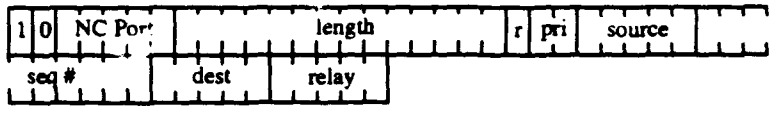
Figure 63 Control message for frame 3 transmission from a non-clusterhead

11.3.4.2 Asynchronous Messages

In the internal network, all asynchronous data traffic is sent in one of three kinds of messages: point-to-point, broadcast, or local messages. A fourth type of message is the null message, which is used to indicate that no more data follows in the slot. This section shows the headers for these messages. Data immediately follows each header, except, of course,.in the case of null messages where no data is present.

11.3.4.2.1 Point-to-Point Asynchronous Message (AMSG_P2P)

Figure 64 shows the point-to-point message header.

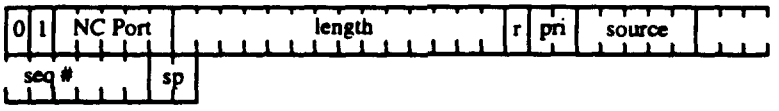


Description of Field	bits
10 - Point-to-Point Message Type	NB_MSG_TYPE
NC_PORT #	NB_NC_PORT
length of msg (bits)	NB_LENGTH
r - reliability	NB_RELIABILITY
pri - priority	NB_PRI
source	NB_NODE_ID
seq # - sequence #	NB_SEQ
dest - destination	NB_NODE_ID
relay	NB_NODE_ID

Figure 64 Point-to-point message header.

11.3.4.2.2 Broadcast Asynchronous Message (AMSG_BRCST)

Figure 65 shows the broadcast message header



Description of Field	bits
01 - Broadcast Message Type	NB_MSG_TYPE
NC_PORT #	NB_NC_PORT
length of msg (bits)	NB_LENGTH
r - reliability	NB_RELIABILITY
pri - priority	NB_PRI
source	NB_NODE_ID
seq # - sequence #	NB_SEQ
sp - spare	fill to nearest 8-bit boundary

Figure 65 Broadcast message header

11.3.4.2.3 Local Asynchronous Message (AMSG_LOCAL)

Figure 66 shows the Local Message header.

Description of Field	bits
00 - Local Message Type	NB_MSG_TYPE
NC_PORT #	NB_NC_PORT
length of msg (bits)	NB_LENGTH
r - reliability	NB_RELIABILITY
sp - spare	fill to nearest 8-bit boundary

Figure 66 Local message header

11.3.4.2.4 Null Asynchronous Message (AMSG_NULL)

Figure 67 shows the Null Message header.

Description of Field	bits
11 - Null Message Type	NB_MSG_TYPE

Figure 67 Null message header

11.3.5 IF_SC_NC

See section 9.3.5.

11.3.6 IF_SC_LC

See section 9.3.4.

11.3.7 IF_OP_SC

See section 9.3.3.

11.4 Notes

None

11.5 Revisions for Section 11

None.

12. NFSE IN-LAB TEST PLAN

12.1 Scope

12.1.1 Identification

This section provides the equivalent of a Software Test Plan (STP) document for the NFSE System.

12.1.2 System Overview

See section 5.1.2 on page 13.

12.1.3 Section Overview

This Software Test Plan describes the formal qualification test plans for all of the CSCIs that make up the NFSE System. The STP identifies the software test environment resources required for formal qualification testing (FQT) and provides schedules for FQT activities. In addition, the STP identifies the individual tests that shall be performed during FQT.

12.2 Referenced Documents

See section 3.

12.3 NFSE In-Lab Test Environment

This subsection identifies and describes the plans for implementing and controlling the resources (software, firmware, and hardware) necessary to perform formal qualification testing. Three test facilities will be used for formal qualification testing. They are identified as the Distributed Simulation and Prototyping Testbed (DSPT), the Realtime Testbed (RTT), and the Target Node Testbed.

Distributed Simulation and Prototyping Testbed (DSPT) - The DSPT is software that is used to simulate and prototype radio communication networks and their protocols. The DSPT software is written in C++ and provides a library of software objects that can be used to construct models of communication systems. For example, the library contains objects to model ships, aircraft, satellites, communication stations, transmitters, receivers, and modems. Simulation channel models are available for HF groundwave and Line-of-Sight (LOS) channels. The DSPT is based on the commercially available simulation package Sim++. This package provides time synchronization for distributed, as well as sequential simulations. All of the network control software for NRL's UNT project was tested using the DSPT, and a simulation model of the Link Controller used in the UNT tests is available.

Realtime Testbed (RTT) - The RTT consists of the hardware needed to perform back-to-back, realtime tests of two NFSE System nodes. Table 28 shows the hardware required for the RTT.

Table 28: Hardware Items for the Realtime Testbed

Item	Description	Qty.
MVME945B-1	VME Benchtop Enclosures	2
MVME135A	CPU (4 Mbyte DRAM)	4
MVME224A-1	Memory (4 Mbyte DRAM)	2
ENP-10L	Ethernet Controllers	2
Sun Sparcstation	Workstation	1

Target-Node Testbed (TNT) - The TNT consists of the target hardware and non-development software for two complete NFSE System nodes including full transmit and receive strings. The two nodes shall be tested back-to-back with connectivity achieved through RF linkage.

12.3.1 Software Items

The following software items are required to support the formal qualification testing activities.

- Distributed Simulation and Prototyping Testbed software.
- Sim++ simulation library and runtime system.
- VxWorks software development environment, including target operating system software.
- C++ compiler.

12.3.1.1 DSPT Software

The purpose of this software is to facilitate the construction of a simulation model of the NFSE System. The DSPT will be used primarily for testing the logical correctness of the algorithms used to implement the Network Frequency Selection Service (NFSS) and the Internal Network Capability (IN_CAP).

12.3.1.2 Sim++

The DSPT is built on top of the commercial simulation package Sim++. Simulation models written in Sim++ will be executed on NRL code 5521's computer workstations.

12.3.1.3 VxWorks

This software provides the cross-development tools and target operating system needed to develop target software on Sun workstations and then move it to the target system and test it.

12.3.1.4 C++ Compiler

The compiler of choice is the Object Center compiler from Center Line.

12.3.2 Hardware and Firmware Items

A Sun SparcStation is required for simulation with the DSPT. The target node hardware, as specified in table 28 is required to support the testing performed using the RTT.

Firmware consists of bootroms on the various boards of the VME system. Bootroms for the MV135 boards will be supplied by Wind Rivers Systems as part of the VxWorks package. Initially, bootroms for the MVME333 boards will be obtained from the UNT Link Controller bootroms used by NRL.

12.3.3 Proprietary Nature and Government Rights

NRL has purchased the right to use Sim++ and VxWorks in the development and in-lab testing of the NFSE System software. Sim++ will not become a part of the target system. Depending on the number of target sites, some additional costs may be required for the use of VxWorks in the target system.

12.3.4 Installation, Testing, and Control

This subsection describes plans for controlling and maintaining each item of the software test environment.

12.3.4.1 DSPT Maintenance

Any maintenance and enhancements required of the DSFT are the primary responsibility of NRL Code 5521.

12.3.4.2 Sim++ Maintenance

NRL will arrange for Jade Simulations Corporation to supply continued maintenance on its Sim++ product.

12.3.4.3 VxWorks Maintenance

NRL will arrange for continued maintenance of the VxWorks system by Wind Rivers Systems for the duration of the NFSE project.

12.3.4.4 Workstation Maintenance

NRL has formal agreements for the maintenance of Code 5521's workstations.

12.3.4.5 Target Hardware Maintenance

TBD

12.3.4.6 MVME135 Bootrom Maintenance

Wind Rivers Systems shall provide and maintain this firmware as part of its VxWorks maintenance agreement with NRL.

12.3.4.7 MVME333 Bootrom Maintenance

NRL Code 5521 shall maintain this firmware.

12.3.4.8 Support Hardware Maintenance

To be determined on a case-by-case basis.

12.4 Formal Qualification Test Identification

12.4.1 NFSE System

The NFSE System CSCIs shall be formally tested in the following combinations: NC, NC/LC, SC/NC, and SC/NC/LC.

12.4.1.1 General Test Requirements

Each formal qualification test shall meet the following general test requirements:

a. There shall be a statement of the purpose of the test; that is, what is to be tested.

b. Test inputs, version numbers for each software item under test, and test results, shall all be archived.

c. Test conclusions shall be reported.

12.4.1.2 Test Classes

Tests are broken up into four classes: algorithm tests, timing tests, start/stop tests, and integration tests.

12.4.1.2.1 Algorithm Tests

These tests are aimed at verification and validation of the system design requirements. That is, these tests attempt to answer the following questions: Does the system that was implemented satisfy the system requirements (verification); and does the system do what the user intends it to do (validation)?

12.4.1.2.2 Timing and Load Tests

These tests are designed to see if the NFSE System software can satisfy its realtime requirements and work within its allocated resources. The main measures are procedure execution times and memory requirements.

12.4.1.2.3 Node Start/Stop/Start Tests

One of the lessons learned from the UNT tests is that one can expect a high occurrence of starting and stopping of nodes in a testbed environment. This class of test is intended as a stress test of the NFSE System's ability to function with frequent occurrences of nodes stopping and restarting.

12.4.1.2.4 Integration Tests

This class of tests focuses on how well the SC, NC, LC, and RF hardware work together in the target system.

12.4.1.3 Test Levels

Formal tests shall be performed at the CSCI, node, or system level. No formal test shall be performed below the CSCI level. However, it is anticipated that considerable informal testing will be done at the CSC/CSU levels.

12.4.1.4 Test Definitions

This subsection identifies and describes each test to be performed on the NFSE System.

12.4.1.4.1 7-Node Simulation (7NFSE)

12.4.1.4.1.1 Test Objective

The purpose of this suite of tests is provide the following: 1) to check out whether the algorithms used to implement the system result in a system that meets the requirements given in the SSRS (verification), 2) to validate that the implemented design meets the intended system needs (validation), and 3) to provide an archive of the performance of the baseline system. The test performance database can be used, among other things, as a benchmark to evaluate alternative metrics that the NFSS might use in selecting the "best" operating frequency.

12.4.1.4.1.2 Special Requirements

These tests will be performed on a Sun workstation or equivalent.

12.4.1.4.1.3 Test Level

The tests will be performed at the NFSE System level using simulated Link Controllers, RF transmit/receive strings, and communication channels. System Mode B (without the optional internal network)

shall be tested. Since the internal network will not be used, the frequency test cycles can be contiguous, i.e., we set $T_{AP} = T_A$ (see figure 3).

12.4.1.4.1.4 Test Type or Class

This is an algorithm test.

12.4.1.4.1.5 Qualification Method

These tests are aimed at demonstrating that the Network Controller can provide the capabilities specified in section 8.1.3.

12.4.1.4.1.6 Type of Data to be Recorded

The following data shall be saved to disk for each test.

- Version identifiers for all of the software being tested and for key support software.
- All input and output data files (as specified in section 10.1.6).

12.4.1.4.1.7 Assumptions and constraints

The tests shall employ realistic channel simulation models that include the effects of stochastic noise and HF groundwave propagation losses.

12.4.1.4.2 Additional Tests

TBD

12.4.1.5 Test Schedule

See section 7.3.2.

12.5 Data Recording, Reduction, and Analysis

This subsection describes the data reduction and analysis procedures to be used during and following the tests identified in this STP. This subsection documents how information resulting from data reduction and analysis will be retained.

Output from the Network Frequency Selection Service are obtained after each NFSS frequency test cycle. These results are in the form of FrameMaker ".mif" files. When viewed using FrameMaker or a PostScript reader, each page of results appears as two columns by four rows of graphics panels. Each panel displays results for one NFSS frequency test cycle. The time-ordering of results is from top to bottom of the first column followed by the same ordering in the second column. Each panel shows the following information: an NFSS frequency test cycle identifier, the bidirectional links for that frequency test cycle, the value of the metric (in hexadecimal notation), the test frequency (in MHz), and the time at which the frequency test cycle began (in simulated hrs:min:sec.ms).

When using the optional internal network, all statistics are merged into a single stat.out file. It is necessary to pass this file through a post-processor filter to generate separate files for each statistic of interest. In this way we produce the following additional output files from stat.out:

load.out
thruput.out
delay.out
max_delay.out
global_traffic_limit.out
lcl_traffic_limit.out

These are in "Cricket Graph" compatible form for automated post analysis, but they can be read manually to get a quick look at network performance. The interpretation of each of these statistics is as follows.

Load Statistic

Every statistic reporting period, each node reports the average rate at which it transmits new traffic since the last reporting period. This traffic consists of either broadcast or point-to-point asynchronous messages. The load includes the message headers added by the Network Controller, which is 40 bits for broadcast messages and 48 bits for point-to-point messages. For each reporting period, the file load.out contains one line of output consisting of the following: the time of the report, the load at each node, and the sum of the loads (i.e., the load on the entire network).

Throughput Statistic

Every statistic reporting period each node reports the average rate at which simulated user-data was delivered at that node since the last reporting period. The "user" data consists of the data-only portion of asynchronous broadcast or point-to-point messages; that is, it does not include the message headers added by the Network Controller. For each reporting period, the file thruput.out contains one line of output consisting of the following: the time of the report, the throughput at each node, the sum of the node throughputs (i.e., the throughput for the entire network), and the "normalized throughput". The *normalized throughput* is obtained by dividing the network throughput by *(A - 1)*, where *A* is the number of nodes active during the reporting interval. If the load consists of only broadcast traffic, then the normalized throughput should be nearly equal to the network load, since each broadcast message should reach *(A -1)* nodes.

Delay Statistic

Every statistic reporting period, each node reports message delay statistics for all the broadcast and point-to-point messages that were received since the last reporting period. This statistic includes minimum, average, and maximum values of the message delays. The delays are measured from the time that the message is first transmitted until the time it is received at a node. This is accomplished by including the initial transmission time in the body of the dummy data. For each reporting period, the file delay.out contains one line of output consisting of the following: the time of the report, the average message delay recorded at each node, and the average message delay for the entire network.

Maximum Delay Statistic

For each reporting period, the file max_delay.out contains one line of output consisting of the following: the time of the report, the maximum message delay recorded at each node for the measurement period, and the maximum message delay for the entire network over that same period.

12.6 Notes

None.

12.7 Revisions to section 12

None.

13. NFSE IN-LAB TEST DESCRIPTIONS

13.1 Scope

13.1.1 Identification

This Software Test Description (STD) describes the in-lab tests to be performed on the Network Controller CSCI of the NFSE System.

13.1.2 System Overview

See section 5.1.2 on page 13.

13.1.3 Section Overview

This section contains the test cases and test procedures necessary to perform formal qualification testing of the Network Controller CSCI, which is the major component of the NFSE System.

13.2 Referenced Documents

See references [3] and [4].

13.3 Formal Qualification Test Preparations

13.3.1 7-Node Simulation (7NFSE)

13.3.1.1 7-Node Simulation Schedule

See section 7.3.2.

13.3.1.2 7-Node Simulation Pre-Test Procedures

13.3.1.2.1 Hardware Preparation

The 7NFSE will be performed on a Sun Workstation or equivalent.

13.3.1.2.2 Software Preparation

In preparation for simulation testing with the DSPT, two input files must be prepared, namely, create.dat and nfss.inp. Examples of these two files are shown in figures 68 and 69, respectively.

File create.dat is used for specifying the simulation scenario. It describes the communication channels, the platforms (ships, aircraft, etc.), and parameters that control the simulation (duration of simulation, statistics reporting period, and period between scenario updates). The file is scanned for the occurrence of object creation commands that contain the keywords create_chan, create, or create_sys. The keyword "create_chan" indicates the start of the description of a communication channel object; the keyword "create"

signifies the start of the description of a platform object; and "create_sys" indicates the description of a system object. The end of an object's description is denoted by the flag 99999. Object descriptions may be nested.

Twelve channels are defined in figure 68. Each channel is an object of type hf_chan. All objects have a type name (hf_chan in this case) and a numerical identifier, which consists of a zero-terminated list of numbers. Thus, in the example there are twelve channel objects of type hf_chan with identifiers that range from 1 to 12. The second line of the create hf_chan entry signifies the median channel noise level in db referenced to 1 watt. In the example, each channel has the same median noise level of -125 db. The final entry in the description of the hf_chan object is the hf frequency of this channel. The frequency is used by the groundwave propagation loss model to compute the propagation attenuation.

The example file create.dat defines seven platforms. Each platform definition begins with the keyword "create". All platforms are of type "near_earth" platform. This type is used to represent fixed-sites, ships, and aircraft. The platforms have numerical identifiers that range from 1 to 7. Platforms have a geographic position, which is expressed in terms of longitude (-180 (W) to 180 (E)), latitude (-90 (S) to 90 (N)), and altitude (0 to 100 km). Since some platforms are mobile, a time (given in seconds) is associated with the position coordinates. Time 0.0 corresponds to the start of the simulation. There are one or more lines of the form "longitude latitude altitude time" included in each platform description. These lines are preceded by a line containing an integer that specifies the number of positions that follow. Platform number 7 is an example of a mobile platform; the other platforms are stationary.

Each platform contains an object of type "logic_node". Logic_nodes have numerical identifiers that consist of two integers - the first number identifies the host platform (1 to 7) and the second identifies the node at that platform (1). Logic_nodes contain objects of types "transmitter", "receiver", "scen_cont", "UNT_PrLayer2", and "UNT_PrLayer3". The last three object types are used to represent the three lowest protocol layers of the communication system that we are modeling.

The start of the description of a transmitter begins with the keyword "transmitter", which is followed by a numerical identifier of the form "platform_number node_number transmitter_number 0". Subsequent lines in the description are: (line 2) channel_name channel_number, (line 3) radiated power (watts), and (line 4) 999999. When the transmitter object is created, it will initially be set to the given channel_number.

The start of the description of a receiver begins with the keyword "receiver", which is followed by a numerical identifier of the form "platform_number node_number receiver_number 0". Subsequent lines in the description are: (line 2) channel_name channel_number, (line 3) required signal-to-noise ratio (db), (line 4) receiver noise (db relative to 1 watt, and (line 5) 999999. When the receiver object is created, it will initially be set to the given channel_number. In the example, each receiver is assigned a receiver noise level of -125 db. The model sets the noise at the receiver as the maximum of the channel noise and the receiver noise. In the example, the required signal-to-noise level of 34 db is meant to approximate a link that is 90% reliable and is operating at 2400 bps.

The start of the description of a protocol object that models protocol_layer_number 1 (physical layer) begins with the keyword "scen_cont", which is followed by a numerical identifier of the form "platform_number node_number protocol_layer_number 0". Subsequent lines in the description are: (line 2) 0 and (line 3) 999999.

The start of the description of a protocol object that models protocol_layer_number 2 (data link layer) begins with the keyword "UNT_PrLayer2", which is followed by a numerical identifier of the form

"platform_number node_number protocol_layer_number 0". Subsequent lines in the description are: (line 2) 0, (line 3) number of transmit strings (1), (line 4) number of receive strings (1), and (line 5) 999999.

The start of the description of a protocol object that models protocol_layer_number 3 (network layer) begins with the keyword "UNT_PrLayer3", which is followed by a numerical identifier of the form "platform_number node_number protocol_layer_number 0". Subsequent lines in the description are: (line 2) 0, (line 3) 1, and (line 4) 999999.

The start of the description of a system object that models the simulator begins with the keyword "system", which is followed by a 0. Subsequent lines in the description are: (line 2) simulation duration (milliseconds), (line 3) statistics reporting period (milliseconds), (line 4) scenario update period (seconds), (line 5) 0, (line 6) 0, and (line 7) 999999.

```
create_chan hf_chan 1 0
-100.000000 db (watts)
10.680 MHz
999999
create_chan hf_chan 2 0
-115.000000 db (watts)
11.062 MHz
999999
create_chan hf_chan 3 0
-135.000000 db (watts)
12.030 MHz
999999
create_chan hf_chan 4 0
-125.000000 db (watts)
12.673 MHz
999999
create_chan hf_chan 5 0
-110.000000 db (watts)
13.238 MHz
999999
create_chan hf_chan 6 0
-125.000000 db (watts)
13.980 MHz
999999
create_chan hf_chan 7 0
-105.000000 db (watts)
14.695 MHz
999999
create_chan hf_chan 8 0
-135.000000 db (watts)
16.923 MHz
999999
create_chan hf_chan 9 0
-120.000000 db (watts)
16.954 MHz
```

```
999999
create_chan hf_chan 10 0
-125.000000 db (watts)
17.514 MHz
999999
create_chan hf_chan 11 0
-130.000000 db (watts)
18.556 MHz
999999
create_chan hf_chan 12 0
-110.000000 db (watts)
20.025 MHz
999999
create near_earth 1 0 NOSC
1
-117.27 32.70 0.0 0.000000
logic_node 1 1 0
transmitter 1 1 1 0
HF 1
40.0 watts
999999
receiver 1 1 1 0
HF 1
34.0 db req S/N
-125.0 db watt noise
999999
scen_cont 1 1 1 0
0
999999
UNT_PrLayer2 1 1 2 0
0
1
1
999999
UNT_PrLayer3 1 1 3 0
0
1
999999
999999
999999
create near_earth 2 0 Point Mugu
1
-119.12 34.12 0.0 0.000000
logic_node 2 1 0
transmitter 2 1 1 0
HF 1
40.0 watts
```

```
999999
receiver 2 1 1 0
HF 1
34.0 db req S/N
-125.0 db watt noise
999999
scen_cont 2 1 1 0
0
999999
UNT_PrLayer2 2 1 2 0
0
1
1
999999
UNT_PrLayer3 2 1 3 0
0
1
999999
999999
999999
create near_earth 3 0 San Clemente Island
1
-118.57 33.03 0.0 0.000000
logic_node 3 1 0
transmitter 3 1 1 0
HF 1
40.0 watts
999999
receiver 3 1 1 0
HF 1
34.0 db req S/N
-125.0 db watt noise
999999
scen_cont 3 1 1 0
0
999999
UNT_PrLayer2 3 1 2 0
0
1
1
999999
UNT_PrLayer3 3 1 3 0
0
1
999999
999999
999999
```

```
create near_earth 4 0 San Nicolas Island
1
-119.52 33.23 0.0 0.000000
logic_node 4 1 0
transmitter 4 1 1 0
HF 1
40.0 watts
999999
receiver 4 1 1 0
HF 1
34.0 db req S/N
-125.0 db watt noise
999999
scen_cont 4 1 1 0
0
999999
UNT_PrLayer2 4 1 2 0
0
1
1
999999
UNT_PrLayer3 4 1 3 0
0
1
999999
999999
999999
create near_earth 5 0 Seal Beach
1
-118.08 33.75 0.0 0.000000
logic_node 5 1 0
transmitter 5 1 1 0
HF 1
40.0 watts
999999
receiver 5 1 1 0
HF 1
34.0 db req S/N
-125.0 db watt noise
999999
scen_cont 5 1 1 0
0
999999
UNT_PrLayer2 5 1 2 0
0
1
1
```

```
999999
UNT_PrLayer3 5 1 3 0
0
1
999999
999999
999999
create near_earth 6 0 Ship
1
-117.84 32.88 0.0 0.000000
logic_node 6 1 0
transmitter 6 1 1 0
HF 1
40.0 watts
999999
receiver 6 1 1 0
HF 1
34.0 db req S/N
-125.0 db watt noise
999999
scen_cont 6 1 1 0
0
999999
UNT_PrLayer2 6 1 2 0
0
1
1
999999
UNT_PrLayer3 6 1 3 0
0
1
999999
999999
999999
create near_earth 7 0 P3
5
-120.00 33.90 1.0 0.0000
-118.00 33.00 1.0 2100.0
-118.00 32.54 1.0 2640.0
-120.00 32.54 1.0 4500.0
-120.00 33.90 1.0 6000.0
logic_node 7 1 0
transmitter 7 1 1 0
HF 1
40.0 watts
999999
receiver 7 1 1 0
```

```
HF 1
34.0 db req S/N
-125.0 db watt noise
999999
scen_cont 7 1 1 0
0
999999
UNT_PrLayer2 7 1 2 0
0
1
1
999999
UNT_PrLayer3 7 1 3 0
0
1
999999
999999
999999
create_sys system 0
1500000.0 //simperiod in simulation time units
259200.0 //report update period in simulation time units
99.75000 //scenario update period in seconds
0
0
999999
```

Figure 68 Example of file create.dat.

Figure 69 contains an example nfss.inp file, which is compatible with the create.dat file shown in figure 68. In order to be compatible, file nfss.inp must have the same number of frequency entries as there are channels in create.dat, and the numbering of the channels in create.dat must correspond to the position of the corresponding frequency in nfss.inp.

```
1.1
10680.0
11062.5
12030.0
12673.5
13238.0
13980.0
14695.0
16923.0
16954.0
17514.0
18556.0
20025.0
-1.0
```

Figure 69 Example of file nfss.inp

13.3.1.2.3 Other Pre-Test Preparations

None.

13.3.2 Additional Tests

TBD

13.4 Formal Qualifications Test Descriptions

13.4.1 7-Node Simulation (7NFSE)

A 7-node simulation model will be used to test the NC CSCI. This model is based on the actual 7-node HF network used for the UNT/NRL HF Intratask Force Communication Network Experiment. This section describes the different test cases for the same simulation model.

13.4.1.1 7-Node Simulation Test 1 (7NFSE-T1)

13.4.1.1.1 7-Node Simulation Test 1 Requirements Traceability

The purpose of this experiment is to verify that the algorithms of the NC are performing properly in an environment of extremely rapidly changing channel noise levels. We show that the algorithms behave as expected. However, we note that the model really precludes the possibility of finding a "best" frequency because performance at every frequency is rapidly and continually changing by significant amounts.

13.4.1.1.2 7-Node Simulation Test 1 Initialization

The NC data files described in Section 10.1.6 are used for this test. The DSPT input file create.dat is used for the specification of the simulation model along with the HF communication channel setup. The NFSS related input file nfss.inp is used for specification of the test frequencies. The initialization data and the formats are discussed in Section 13.4.1.1.3

The DSPT based simulated environment is created using the input file create.dat. This file uses a pre-specified format for the description of the 7-node model and the HF channels for the communication network. The file format is described in the above referenced documents. The HF channel specifications, for the NFSE test channels, are discussed in Section 13.4.1.1.3.

13.4.1.1.3 7-Node Simulation Test 1 Inputs

For this test case, we use the set of 12 frequencies 12161, 12162, ... , 12172 kHz, which correspond to channels 1 through 12, respectively. The slightly different frequencies are used for identification purposes only. Each channel has the same median noise level of -130 db. Each receiver is assumed to have a receiver noise level of -150 db. All platforms are assumed to be stationary. Other input parameters are as shown in figure 68.

The noise model was modified for this test to produce a sinusoidally time-varying about the constant median level. The amplitude of the sinusoid is set at 10% of the median noise level. All channels share the same time-varying model. The channel noise levels and the receiver noise levels are chosen in such a way that the channel noise dominates. Thus, performance on a given frequency depends on when the channel is tested.

The sinusoidal time-period for the channel noise is set at 10 minutes. The duration of each frequency test cycle is 99,750 ms = (3epochs* 5 frames/epoch +1epoch*10frames/epoch)* 7slots/frame * 570ms/slot. Based on a 10 minute time period for the sinusoidal noise variation, there will be approximately 6 such test cycles in one sinusoidal time-period. During each test cycle, three new frequencies plus the current "best" frequency are tested.

13.4.1.1.4 7-Node Simulation Test 1 Expected Results

Figure 70 depicts an example test sequence to illustrate the expected results. For illustration purposes we assume a linear noise increase. For Test Cycle i the four test frequencies are evaluated at the time instants marked 1 through 4. The best metric, for identical receiver noise levels, will correspond to the first frequency (say f1). In Test Cycle i+1, the best frequency carried over from previous cycle will be f1. The metric evaluated for f1 in Test cycle i+1 will be, in general, greater than the one from Test Cycle i. In test Cycle i+1 the best frequency metric will correspond to time instant5 (with corresponding frequency denoted as f5). Thus the best frequency, for monotonically increasing noise levels and same receiver noise, will be the first test frequency during each test cycle. In an analogous manner, for decreasing noise case, the best frequency will always be the last test frequency, i.e. the best frequency from the previous cycle. Thus decreasing noise levels will always result in a the selection of the same "best" frequency for a series of test cycles.

Figure 71 summarizes the expected results for the sinusoidal noise model. Note that there is a phase shift in the sinusoid since, in the actual tests, the first NFSS test cycle does not occur until 79.8 seconds into the simulation. Figure 71 shows the expected sequence of frequencies tested (and therefore the expected sequence of best frequencies, as well). This is indicated by the channel number that is written in

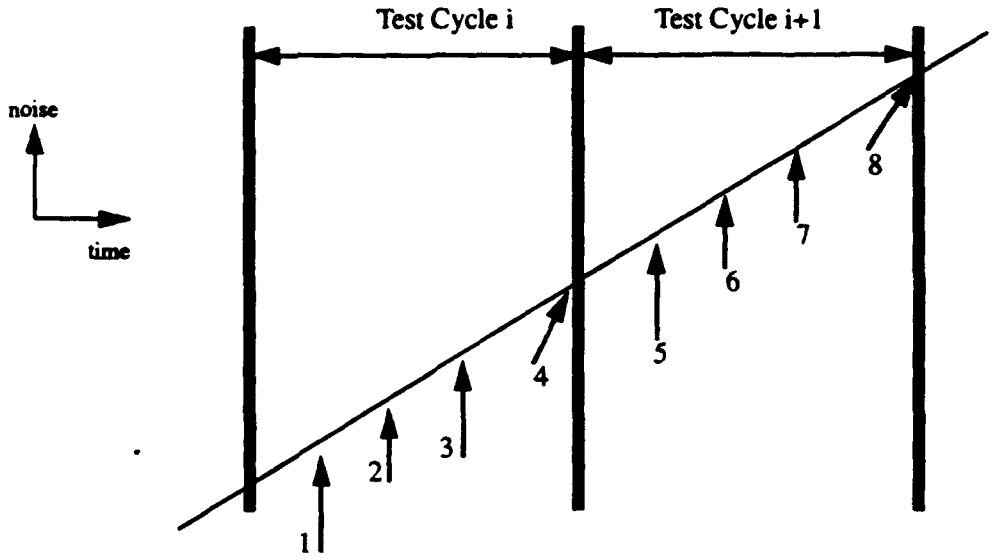


Figure 70 Test cycle sequences during a period of increasing noise levels..

each frequency test cycle for the given sinusoidal noise model. The anticipated channel sequence was generated on the basis of the discussion in the previous paragraph. For example, for increasing noise levels, the first test frequency in each test cycle is expected to be the best frequency. In an analogous manner, for decreasing noise levels the best frequency corresponds to the last frequency. Hence, over periods of decreasing noise, the same frequency is always selected as the best frequency.

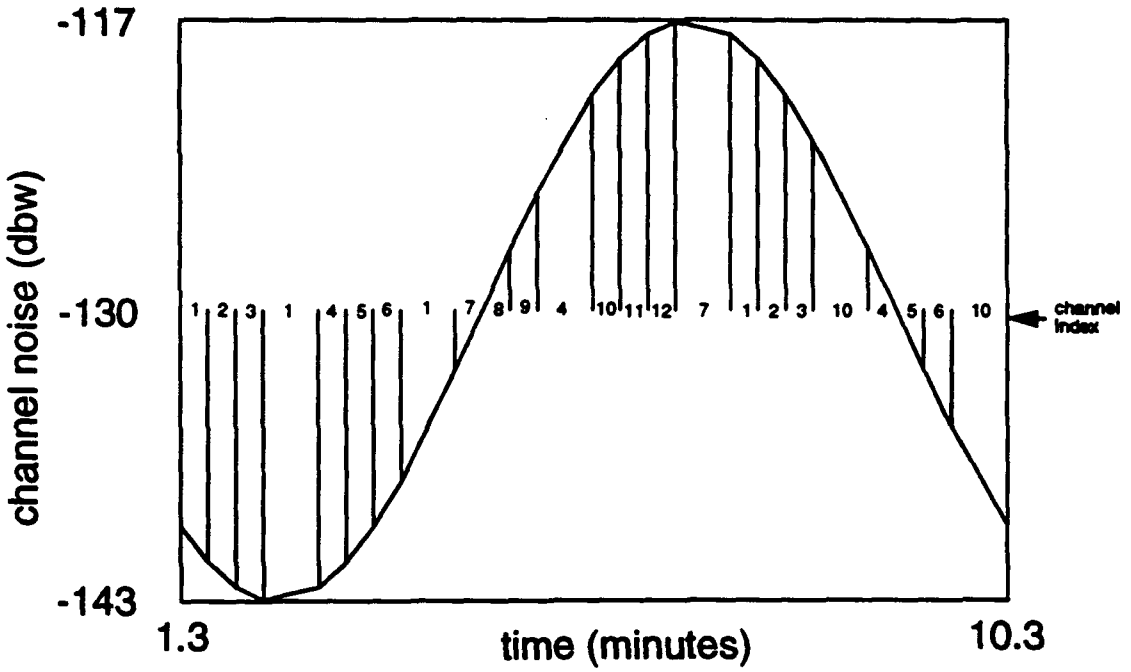


Figure 71 Anticipated results for Test 1 based on relative noise levels of the tested channels.

13.4.1.1.5 7-Node Simulation Test 1 Criteria for Evaluating Results

The simulation results should approximate the expected results discussed in section 13.4.1.1.4.

13.4.1.1.6 7-Node Simulation Test 1 Procedure

The test is to be conducted as a Sim++ simulation running on a Sun workstation.

13.4.1.1.7 7-Node Simulation Test 1 Assumptions and Constraints

See section 13.4.1.1.3.

13.4.1.2 7-Node Simulation Test 2 (7NFSE-T2)

This test case uses frequencies that were allocated for the UNT test. The 7-node model for Test 2 has stationary nodes. The noise levels for the HF channels are constant.

13.4.1.2.1 7-Node Simulation Test 2 Requirements Traceability

This test is used to verify the operation of the network frequency selection algorithm under static conditions. This test provides a general test case and serves as a benchmark for comparing different metrics.

13.4.1.2.2 7-Node Simulation Test 2 Initialization

See Section 13.4.1.1.2.

13.4.1.2.3 7-Node Simulation Test 2 Inputs

As described in Section 13.4.1.1.3, the NFSE specific test inputs are the test frequency channels along with their associated noise levels. Table 30 shows the different frequencies, their associated channel noise levels, and the corresponding communication ranges used for this test case. The receiver noise is taken to be -125 db. All platforms are assumed to be stationary. All other values are as given in figure 68.

Table 29: Frequencies, channel noise levels, and communication ranges for tests 7NFSE-T2.

Freq. Array Index	Freq.(Khz)	Channel Noise(db)	Range[a] (Km)
0	10680.0	-100.0	37.97
1	11062.5	-115.0	81.72
2	12030.0	-135.0	115.38
3	12673.5	-125.0	109.59
4	13238.5	-110.0	50.0
5	13980.0	-125.0	98.6

Table 29: Frequencies, channel noise levels, and communication ranges for tests 7NFSE-T2.

Freq. Array Index	Freq.(Khz)	Channel Noise(db)	Range[a] (Km)
6	14695.0	-105.0	36.6
7	16923.0	-135.0	77.35
8	16954.0	-120.0	60.35
9	17514.0	-125.0	73.57
10	10315.0	-130.0	132.48
11	8578.0	-110.0	110.96

a. Range is based on the noise at the receiver, which is approximated as the larger of the channel noise (see column 3) and the receiver noise (-125 db).

13.4.1.2.4 7-Node Simulation Test 2 Expected Results

We expect that the channel with the greatest communication range will be selected as the "best" frequency. For this test case, with 12 test frequencies, which are tested at the rate of 3 new frequencies per test cycle, it takes 4 Test Cycles (or 399 seconds) to complete the testing of all frequencies. Thereafter, the results of tests at a given frequency should simply repeat the earlier results.

13.4.1.2.5 7 Node Simulation Test 2 Criteria for Evaluating Results

Each test cycle result can be evaluated individually on the basis of the frequencies tested and their corresponding HF communication ranges.

13.4.1.2.6 7 Node Simulation Test 2 Procedure

See section 13.4.1.1.6.

13.4.1.2.7 7 Node Simulation Test 2 Assumptions and Constraints

See section 13.4.1.2.3.

13.4.1.3 7-Node Simulation Test 3 (7NFSE-T3)

This test case uses the same setup at 7NFSE-T2 (See section 13.4.1.2)- the only difference being that it uses a dynamic, 7- node model, i.e., during the simulation run one of the nodes traverses a pre-defined path through the remaining nodes.

13.4.1.3.1 7-Node Simulation Test 3 Requirements Traceability

This test verifies the network frequency selection algorithm under changing connectivities induced by the physical model dynamics. The noise levels for the HF channels are still kept constant, with the values shown in table 30.

13.4.1.3.2 7-Node Simulation Test 3 Initialization

See section 13.4.1.1.2.

13.4.1.3.3 7-Node Simulation Test 3 Inputs

Table 30 shows the different frequencies and their associated channel noise levels used for this test case.

Table 30: Frequencies, channel noise levels, and communication ranges for tests 7NFSE-T3.

Freq. Array Index	Freq.(Khz)	Channel Noise(db)	Range[a] (Km)
0	10680.0	-100.0	37.97
1	11062.5	-115.0	81.72
2	12030.0	-135.0	115.38
3	12673.5	-125.0	109.59
4	13238.5	-110.0	50.0
5	13980.0	-125.0	98.6
6	14695.0	-105.0	36.6
7	16923.0	-135.0	77.35
8	16954.0	-120.0	60.35
9	17514.0	-125.0	73.57
10	18556.0	-130.0	67.22
11	20025.0	-110.0	29.75

a. Range is based on the noise at the receiver, which is approximated as the larger of the channel noise (see column 3) and the receiver noise (-125 db).

13.4.1.3.4 7-Node Simulation Test 3 Expected Results

See section 13.4.1.2.4.

13.4.1.3.5 7 Node Simulation Test 3 Criteria for Evaluating Results

See section 13.4.1.2.5.

13.4.1.3.6 7 Node Simulation Test 3 Procedure

See section 13.4.1.1.6.

13.4.1.3.7 7 Node Simulation Test 3 Assumptions and Constraints

See section 13.4.1.2.3.

13.4.1.4 Other Tests

TBD

13.5 Notes

None.

13.6 Revisions to Section 13

None.

14. NFSE IN-LAB TEST REPORTS

14.1 Section Overview

This section contains the test results of formal qualification testing of the NC CSCI of the NFSE System. During these tests we were particularly interested in testing the network frequency selection algorithm.

14.2 Referenced Documents

See references [3] and [4].

14.3 Test Overview

14.3.1 7-Node Simulation (7NFSE) Test Overview

14.3.1.1 7-Node Simulation Summary

The 7-node simulation model is derived from the actual 7-node HF Network used for the UNT/ NRL HF Intratask force communication network experiment. The system initialization and control parameters were set on the basis of that network. Three different tests were performed for verifying and validating the network frequency selection scheme. On the basis of the input parameter settings and the expected heuristic system behavior, the results were carefully analyzed and the correctness of the results were checked. Based on the results obtained, minor improvements in the metric definition/selection were made (see section 10.3.8.1.

14.3.1.2 7-Node Simulation Record

The in-lab tests were performed at the Naval Research Laboratory, Washington DC, in July 1993.

14.4 Test Results

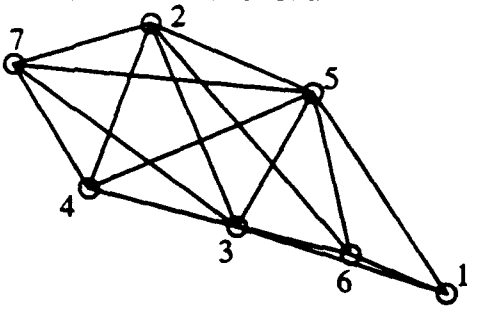
14.4.1 7-Node Simulation (7NFSE) Results

14.4.1.1 7-Node Simulation Test 1 Results

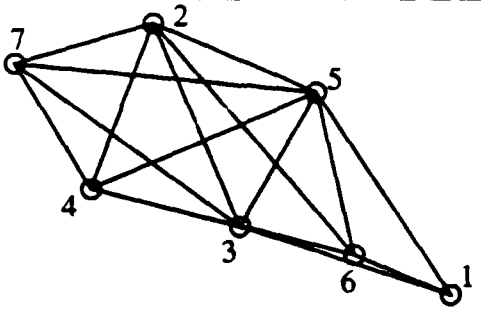
Figures 72 through 74 present the results of the first 12 NFSS test cycles of a simulation run. The 7NFSE-T1 test was performed using a sinusoidal noise model, with a time-period of 10 minutes. As each NFSS test cycle duration is 99.75 seconds, a single noise cycle corresponds to approximately 6 NFSS test cycles. The individual frequency tests (total 4) in each NFSS test cycle are presented vertically with two NFSS test cycles per figure. Hence figures 72 through 74 represent results for two noise cycles. Each panel has an identifier that is coded as XX.Y, where XX is the NFSS test cycle number and Y is the frequency test cycle within a given NFSS test cycle. Simulation time appears as hours:min:sec.ms in each panel. The remaining items in each panel are self-explanatory. The results presented are for the case with channel noise set to -130 db with a 10% sinusoidal variation. Since the receiver noise was kept constant at -150 db, this provided a receiver-noise independent NFSE testing setup.

Test 1 results shown in figures 72 through 74 differ slightly from the expected connectivity-range-based results shown in figure 71. The actual results show the sequence (1, 2, 3, 1), (4, 5, 6, 1), (7, 8, 9, **1**), (10, 11, 12, 7), (1, 2, 3, 10), (4, 5, 6, **3**), ... The numbers in bold face indicate where the anticipated results differ from the simulation results. On closer inspection of the results, the deviation can be attributed to the way Metric A resolves ties between topologies that have the same number of disconnected nodes and the same number of relay nodes. The initial metric (referred to as Metric A) resolves ties in favor of the frequency with the lower frequency (channel) index. Hence, in the second NFSS test cycle (Nfss tests 2.1 through 2.4, which appear in figure 72), each frequency that was tested required one relay node. However, frequency channel 1 (12.161 MHz) was selected because it had the lowest index (0) in the frequency table, although other frequencies clearly provided more links. Similar results occur in NFSS tests 5.1 through 5.4.

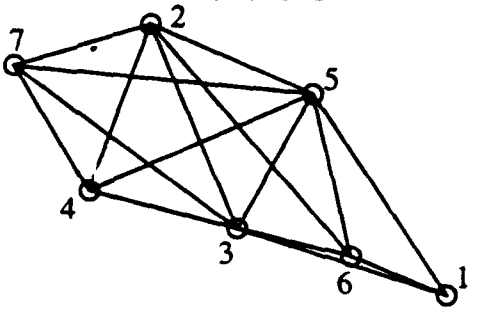
In order to take the overall network connectivity into account, the NFSS was tested using a new metric (referred to as Metric B, and defined in section 10.3.8.1), which refines the frequency selection process to also consider the number of links in the network. Figures 75 through 77 show the results for Metric B. Here the resulting frequency selection sequence does match the expected sequence shown in figure 71. Subsequent tests were all performed using Metric B.



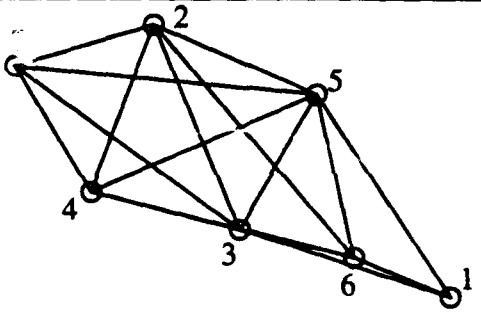
Nfss Test 1.1; Metric=0x40
12.1610Mhz; (0:01:19.800)



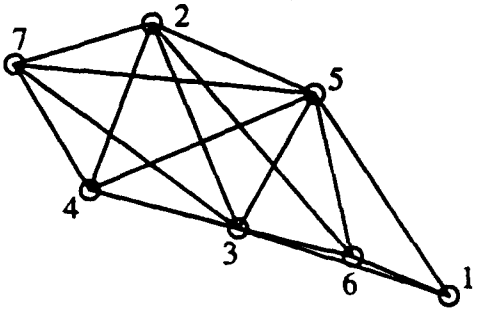
Nfss Test 2.1; Metric=0x43
12.1640Mhz; (0:02:59.550)



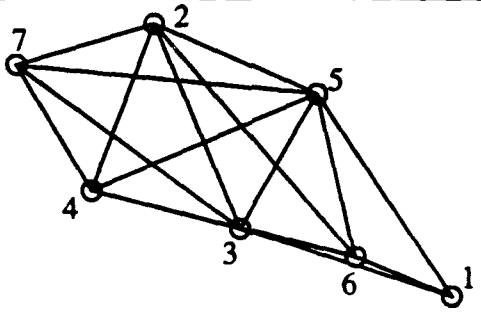
Nfss Test 1.2; Metric=0x41
12.1620Mhz; (0:01:39.750)



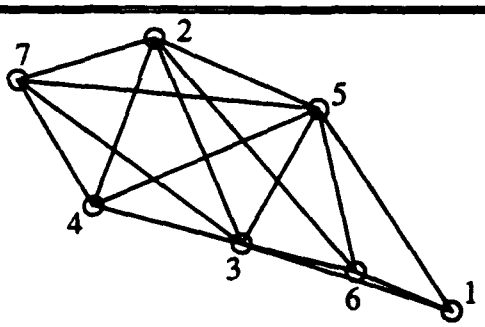
Nfss Test 2.2; Metric=0x44
12.1650Mhz; (0:03:19.500)



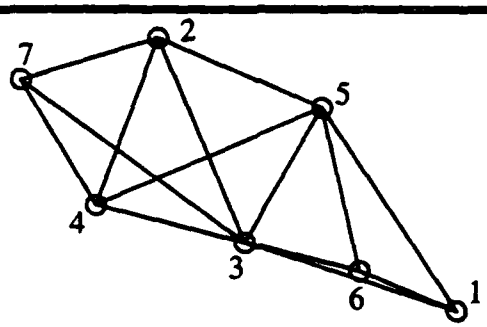
Nfss Test 1.3; Metric=0x42
12.1630Mhz; (0:01:59.700)



Nfss Test 2.3; Metric=0x45
12.1660Mhz; (0:03:39.450)

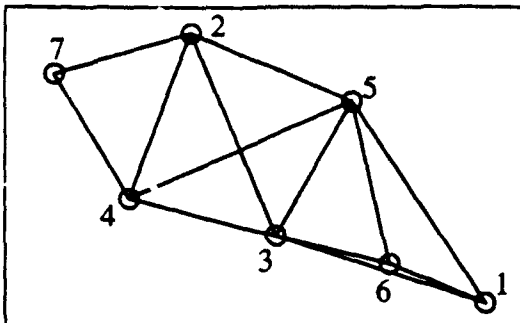


Nfss Test 1.4; Metric=0x40
12.1610Mhz; (0:02:19.650)

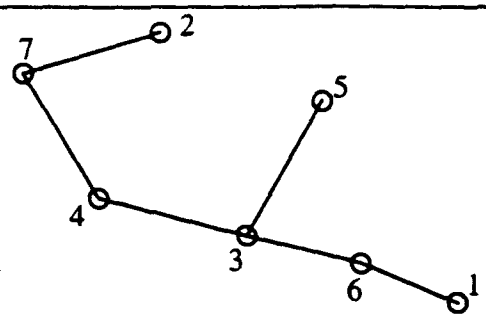


Nfss Test 2.4; Metric=0x40
12.1610Mhz; (0:03:59.400)

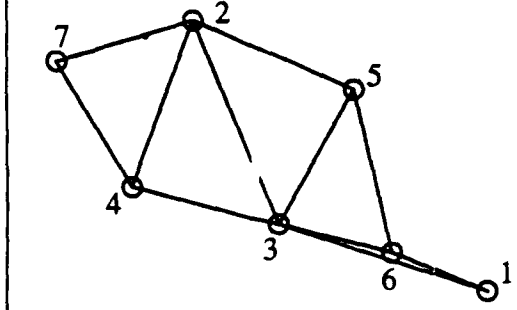
Figure 72 Test 1 (metric A) Results: Set 1



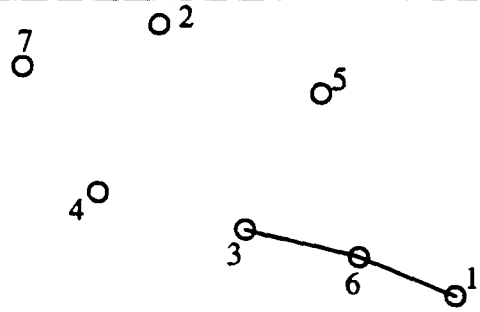
Nfss Test 3.1; Metric=0x86
12.1670Mhz; (0:04:39.300)



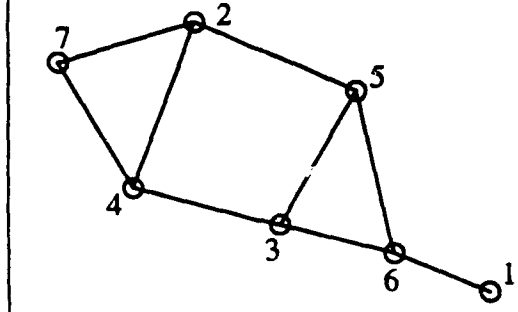
Nfss Test 4.1; Metric=0x109
12.1700Mhz; (0:06:19.050)



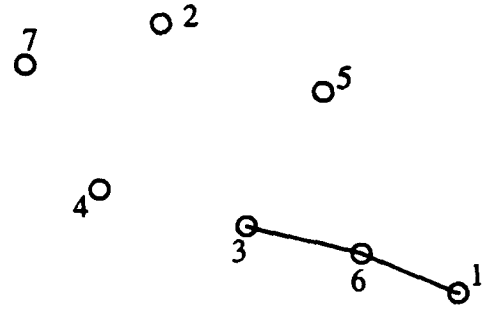
Nfss Test 3.2; Metric=0x87
12.1680Mhz; (0:04:59.250)



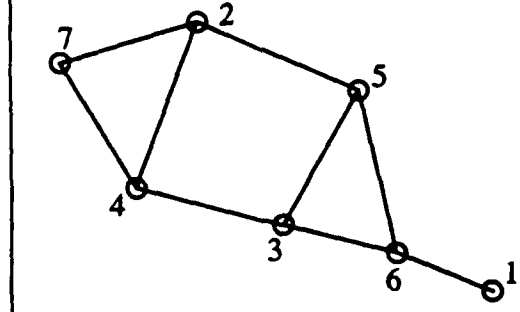
Nfss Test 4.2; Metric=0x204a
12.1710Mhz; (0:06:39.000)



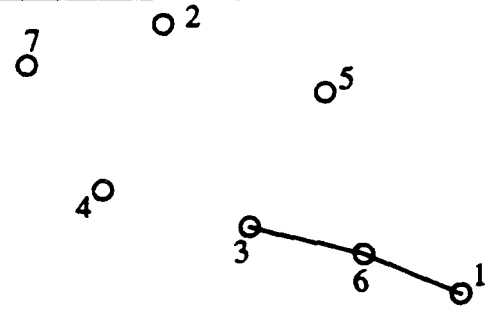
Nfss Test 3.3; Metric=0xc8
12.1690Mhz; (0:05:19.200)



Nfss Test 4.3; Metric=0x204b
12.1720Mhz; (0:06:58.950)

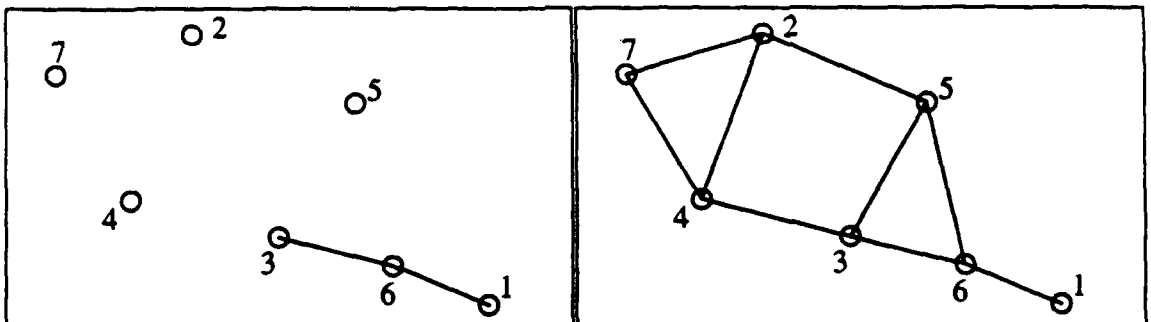


Nfss Test 3.4; Metric=0xc0
12.1610Mhz; (0:05:39.150)



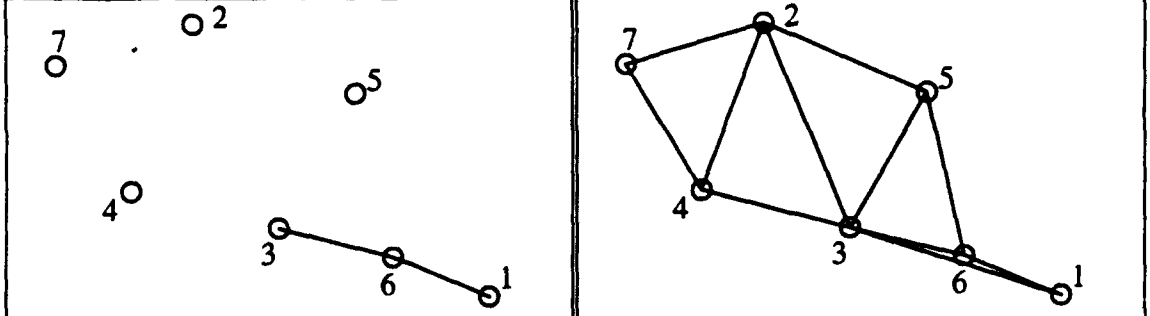
Nfss Test 4.4; Metric=0x2046
12.1670Mhz; (0:07:18.900)

Figure 73 Test 1 (metric A) Results: Set 2



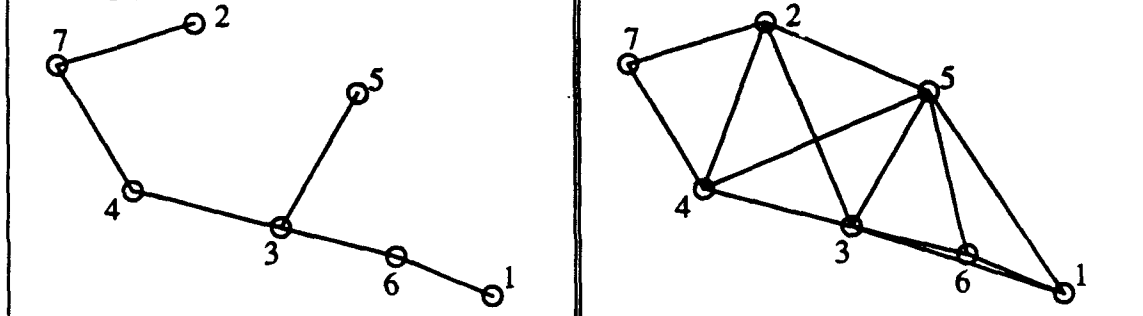
Nfss Test 5.1; Metric=0x2040
12.1610Mhz; (0:07:58.800)

Nfss Test 6.1; Metric=0xc3
12.1640Mhz; (0:09:38.550)



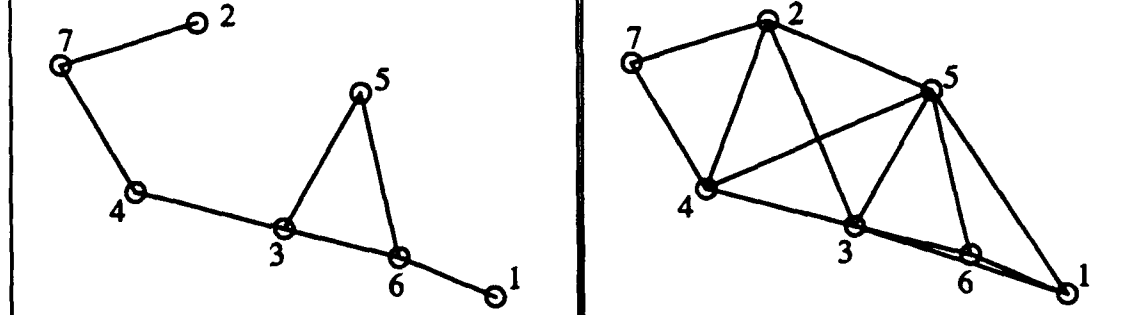
Nfss Test 5.2; Metric=0x2041
12.1620Mhz; (0:08:18.750)

Nfss Test 6.2; Metric=0x84
12.1650Mhz; (0:09:58.500)



Nfss Test 5.3; Metric=0x102
12.1630Mhz; (0:08:38.700)

Nfss Test 6.3; Metric=0x85
12.1660Mhz; (0:10:18.450)



Nfss Test 5.4; Metric=0x109
12.1700Mhz; (0:08:58.650)

Nfss Test 6.4; Metric=0x82
12.1630Mhz; (0:10:38.400)

Figure 74 Test 1 (metric A) Results: Set 3

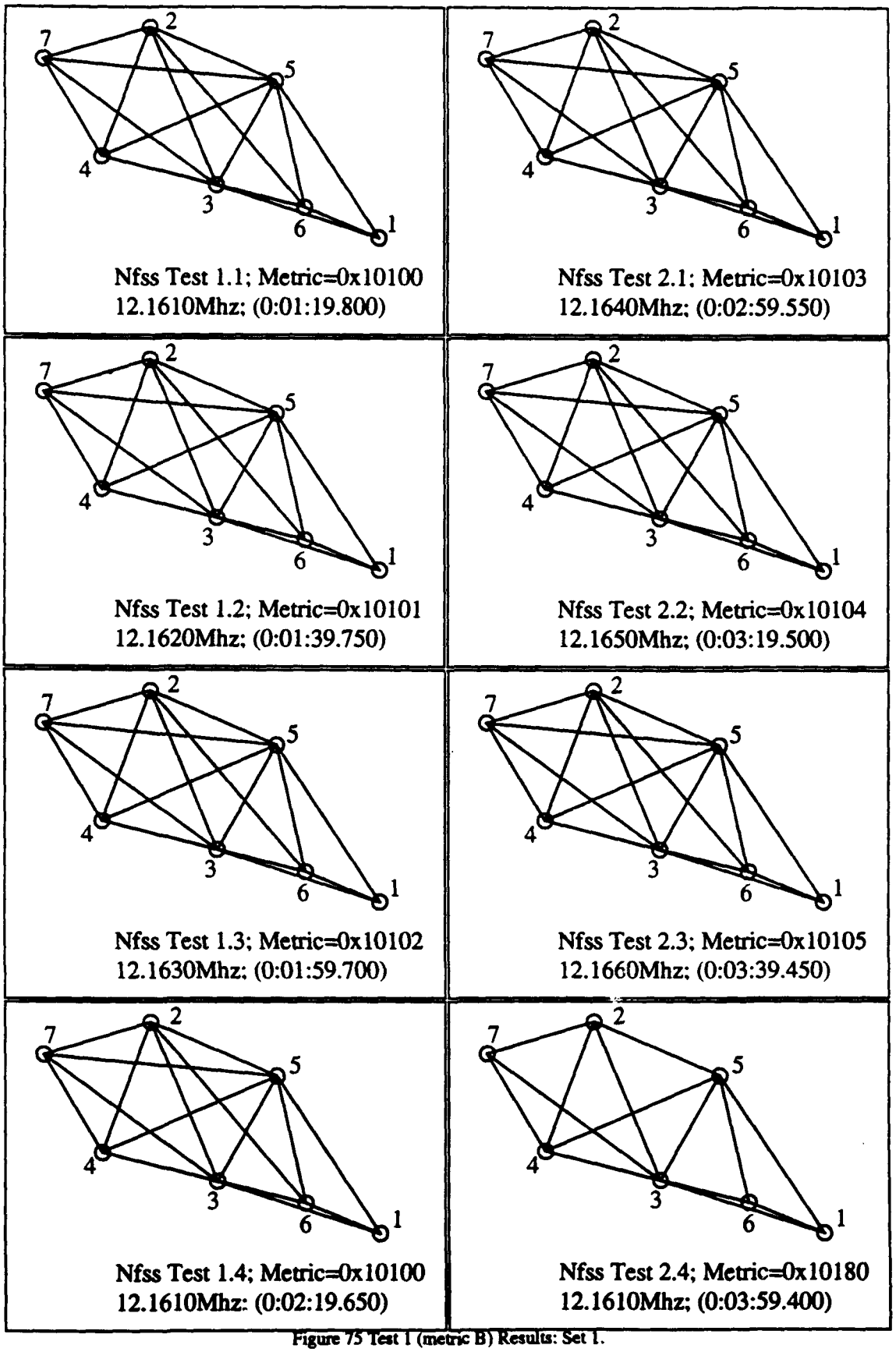


Figure 75 Test 1 (metric B) Results: Set 1.

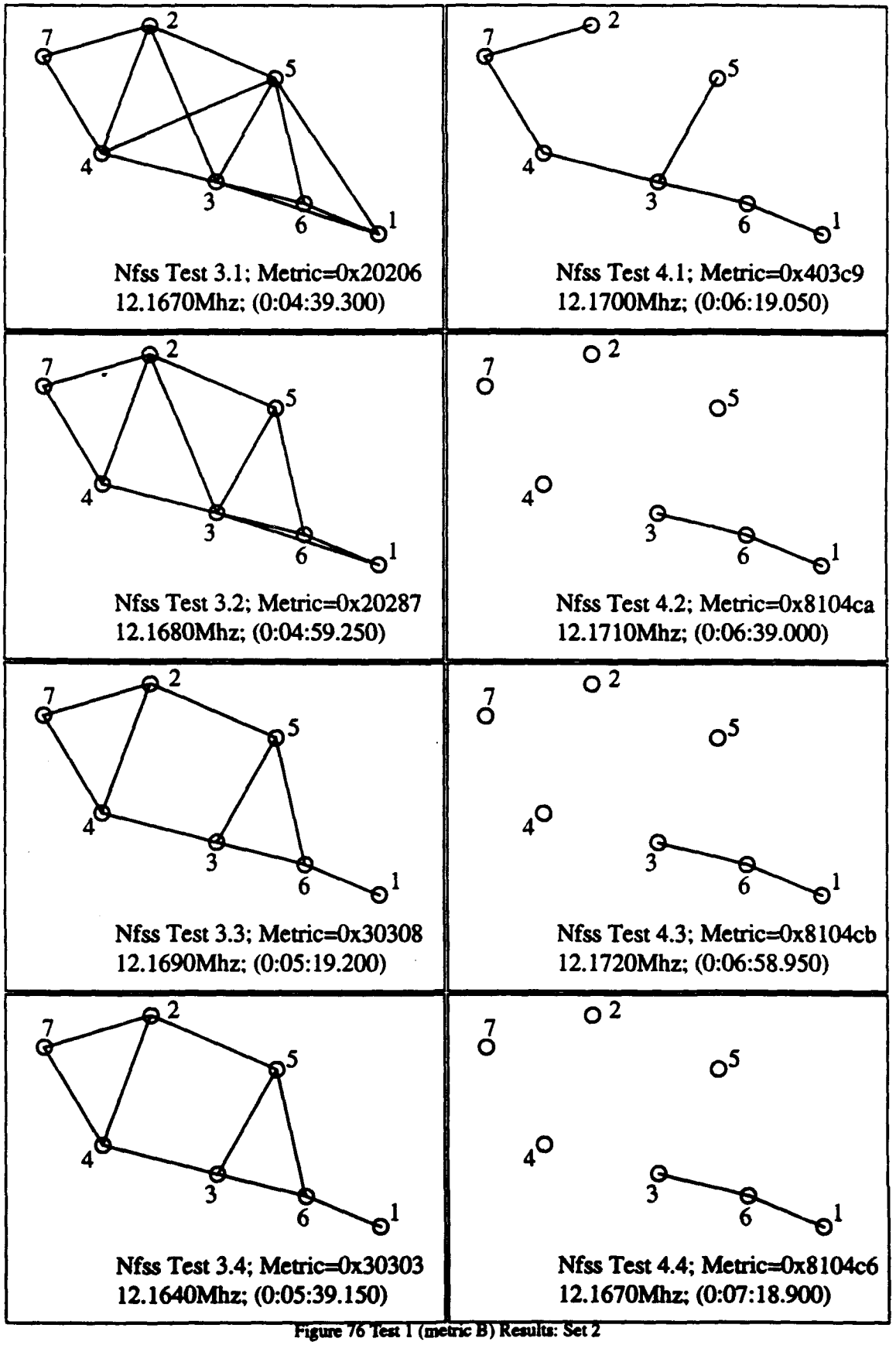


Figure 76 Test 1 (metric B) Results: Set 2

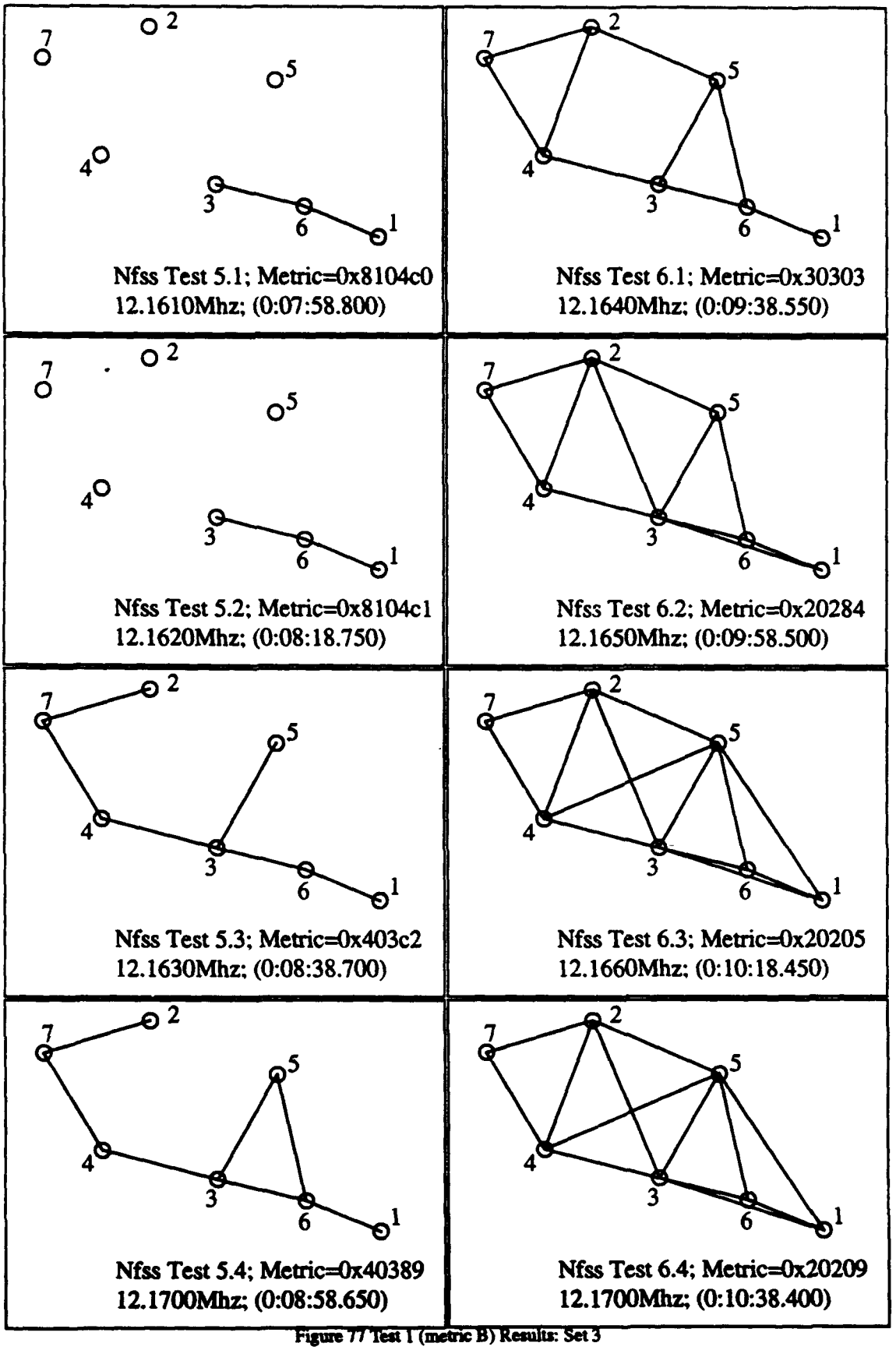


Figure 77 Test 1 (metric B) Results: Set 3

14.4.1.2 7 Node Simulation Test 2 results

As mentioned in section 13.4.1.2.4, we expect that, for this example, the best frequency will be the one with the greatest communication range. For Test 2, the frequencies and communication ranges are shown in table 30. Figures 78 and 80 show Test 2 results for 24 frequency test cycles (more than enough to cycle through the entire set of frequencies). As expected the best metric corresponds to test frequency 10.31 Mhz with an associated communication range of 132 km. The best frequency for NFSS test cycles 1, 2 and 3 is 12.03 Mhz (with range 115 kms), but for NFSS test cycle 4 the best frequency is updated to its final value of 10.315 Mhz.

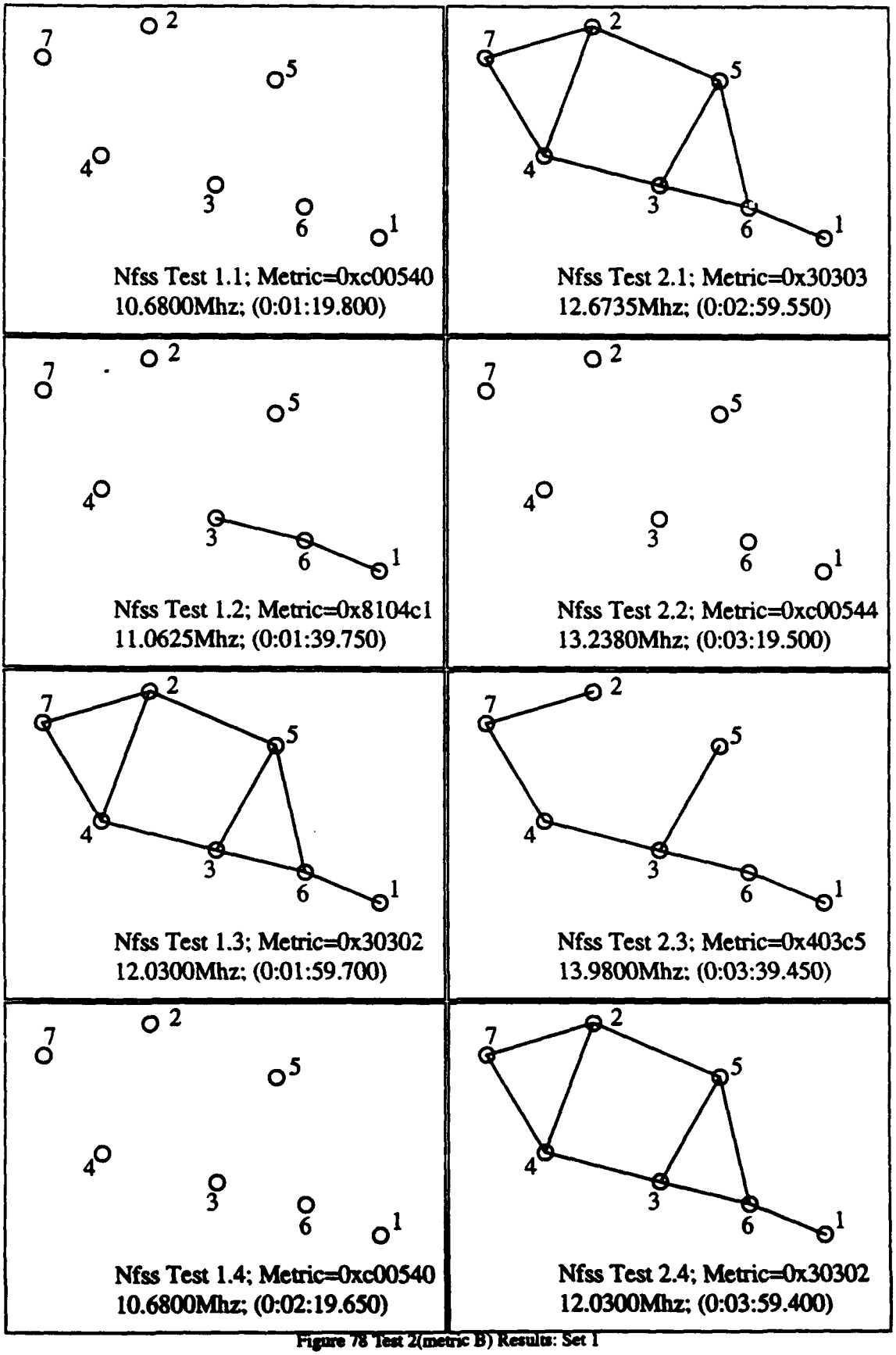


Figure 78 Test 2(metric B) Results: Set 1

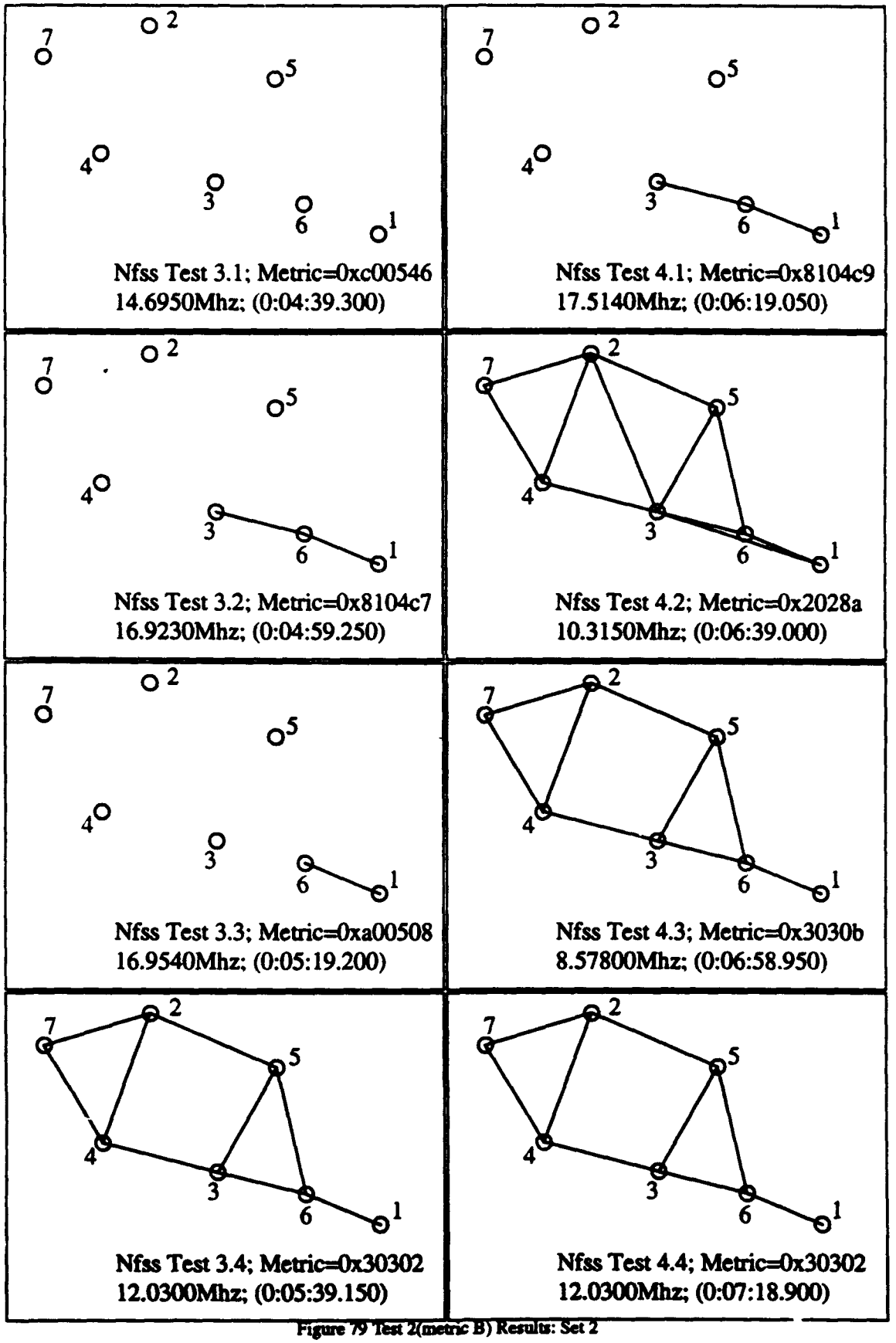


Figure 79 Test 2(metric B) Results: Set 2

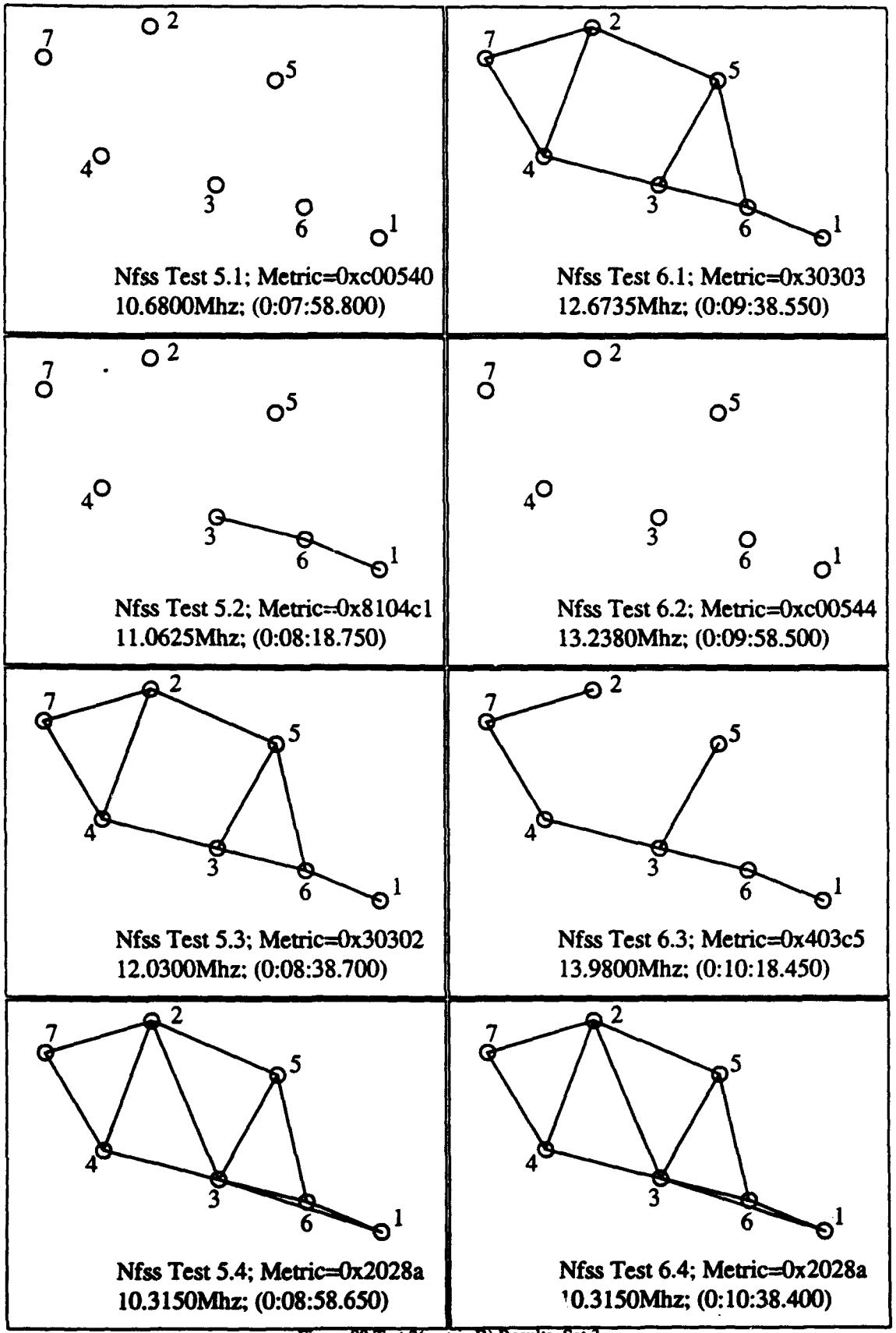


Figure 80 Test 2(metric B) Results: Set 3

14.4.1.3 7 Node Simulation Test 3 results

The 7NFSE-T3 test incorporates a model with one mobile node. Platform 7 was a simulated as a dynamic node, which traversed a pre-specified, deterministic path through the remaining nodes. This allowed for a realistic representation of an actual HF network and demonstrates the effects of platform motion on the frequency selection process. The results are shown in figures 81 through 86. As predicted in section 13.4.1.3.4, after all frequencies have been tested, the best frequency corresponds to the channel with the greatest communication range. For Test 3, channel 3, with frequency 12.03 MHz, has the greatest communication range (115 km). The results show that this frequency is eventually chosen as the best frequency. The results also illustrate how the values of the metric change as platform 7 moves. For example, at the frequency 12.03 MHz the metric changes from 0x30302 to 0x302c2 to 0x30282 to 0x20242 as platform 7 moves into positions where more links are formed.

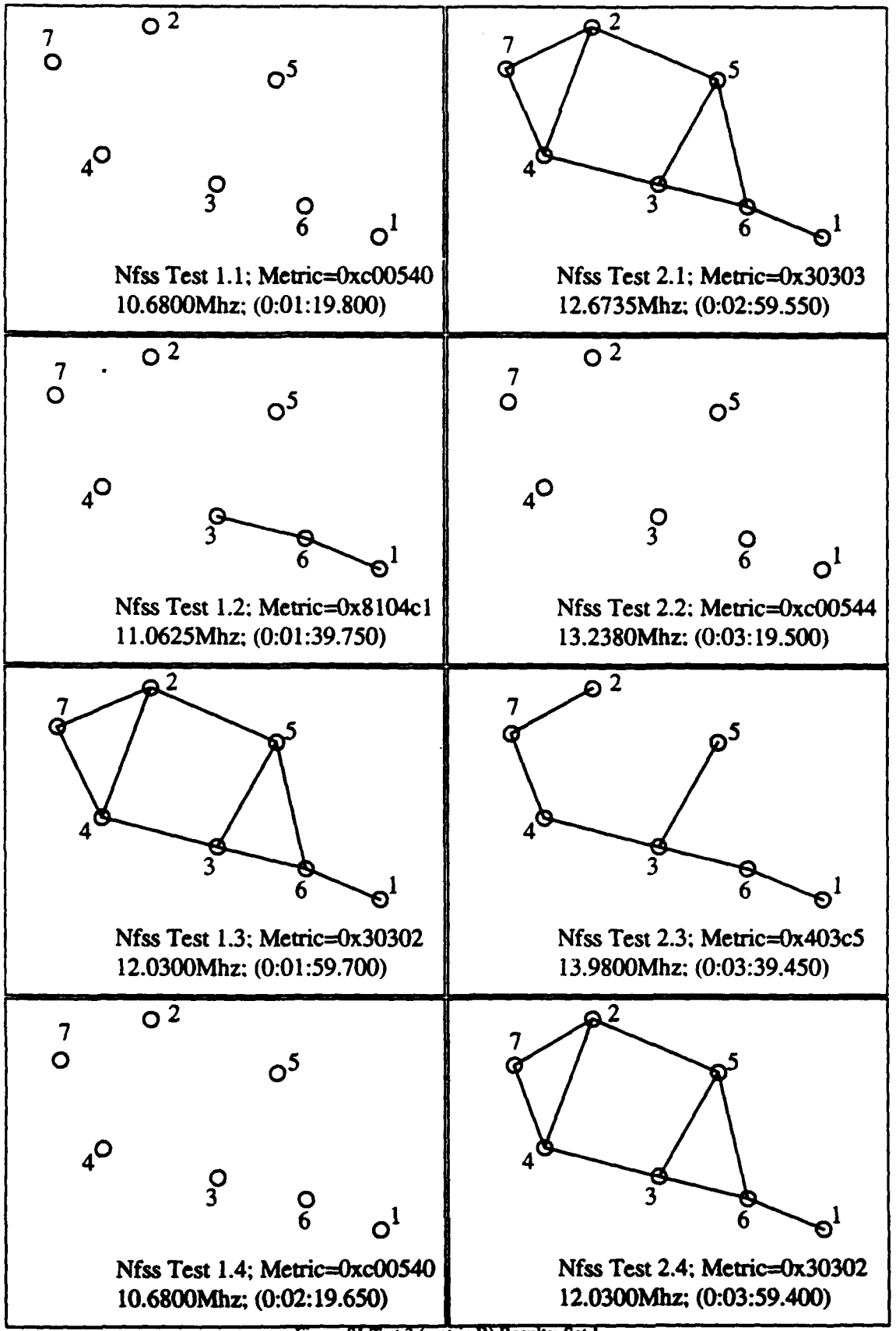


Figure 81 Test 3 (metric B) Results: Set 1

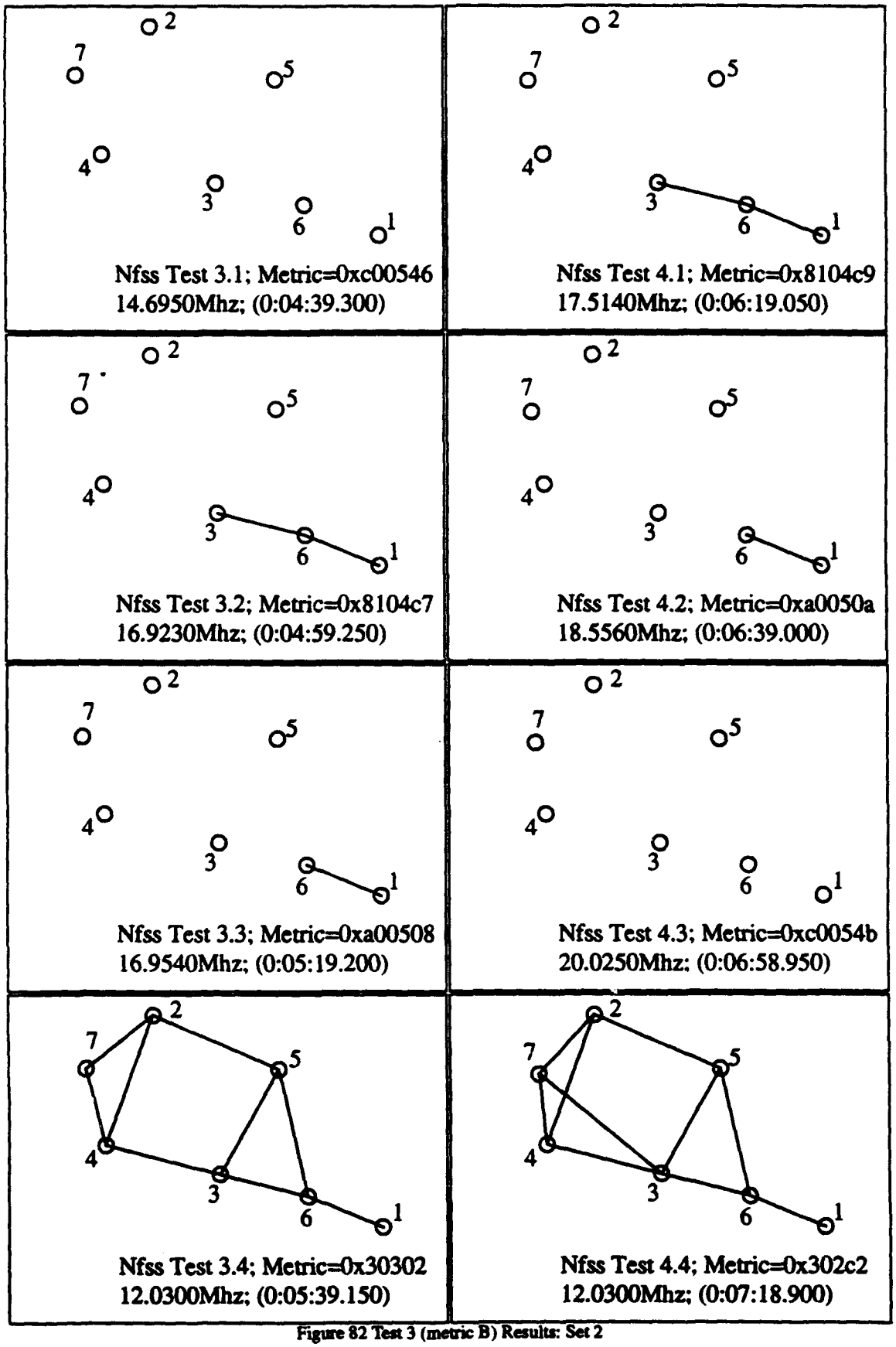
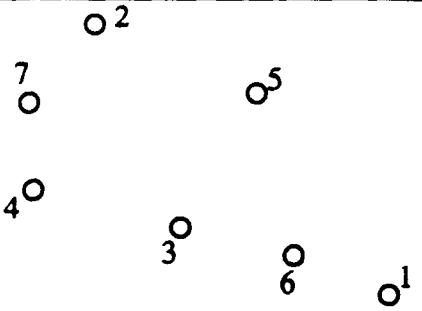
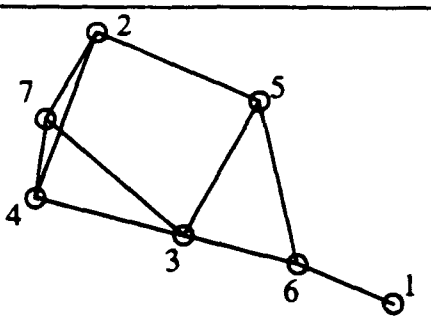


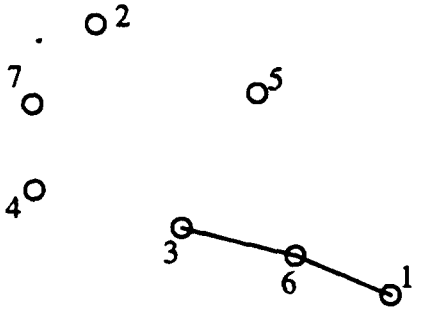
Figure 82 Test 3 (metric B) Results: Set 2



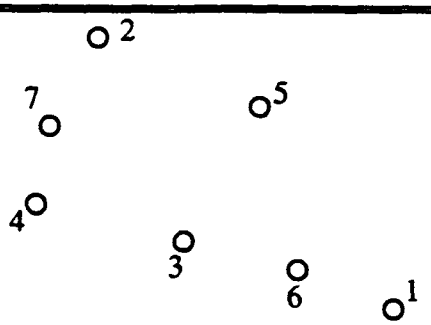
Nfss Test 5.1; Metric=0xc00540
10.6800Mhz; (0:07:58.800)



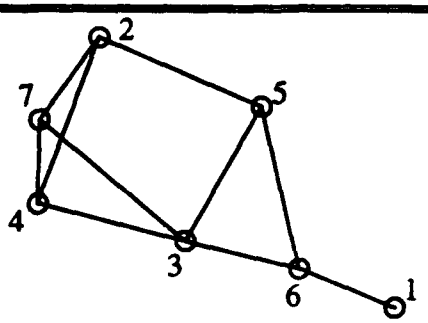
Nfss Test 6.1; Metric=0x302c3
12.6735Mhz; (0:09:38.550)



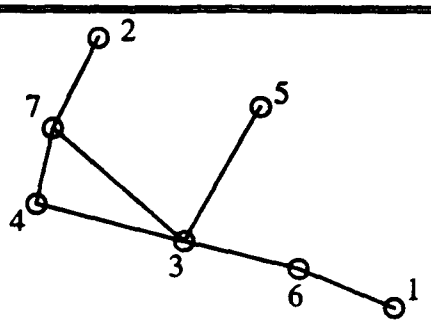
Nfss Test 5.2; Metric=0x8104c1
11.0625Mhz; (0:08:18.750)



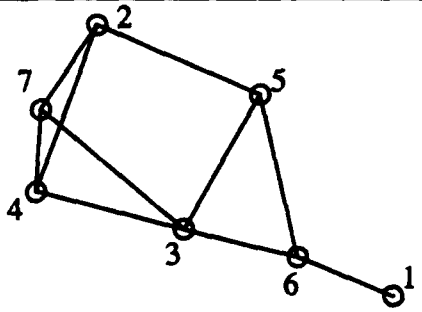
Nfss Test 6.2; Metric=0xc00544
13.2380Mhz; (0:09:58.500)



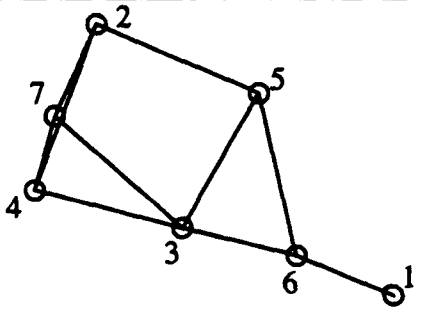
Nfss Test 5.3; Metric=0x302c2
12.0300Mhz; (0:08:38.700)



Nfss Test 6.3; Metric=0x30385
13.9800Mhz; (0:10:18.450)



Nfss Test 5.4; Metric=0x302c2
12.0300Mhz; (0:08:58.650)



Nfss Test 6.4; Metric=0x302c2
12.0300Mhz; (0:10:38.400)

Figure 83 Test 3 (metric B) Results: Set 3

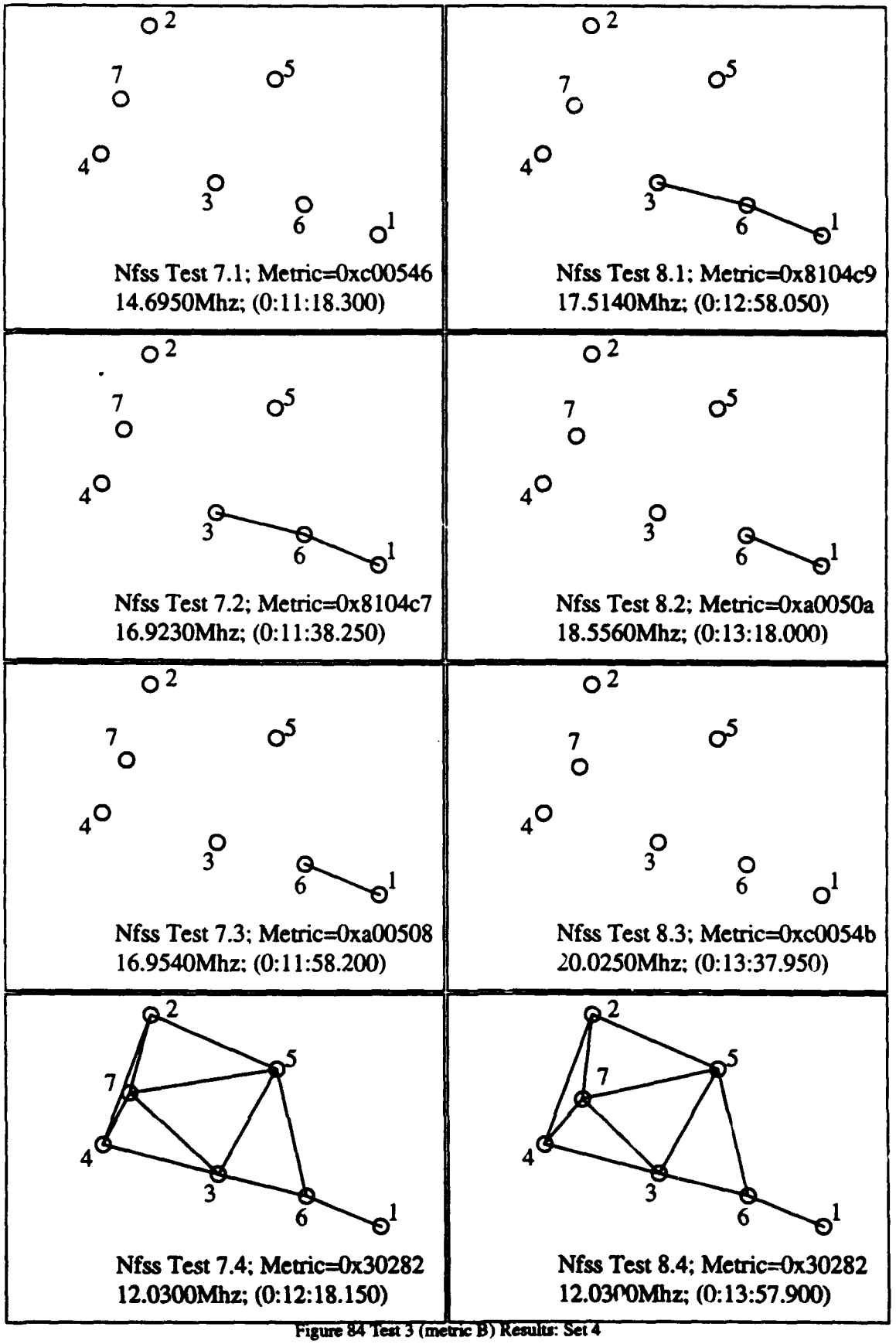


Figure 84 Test 3 (metric B) Results: Set 4

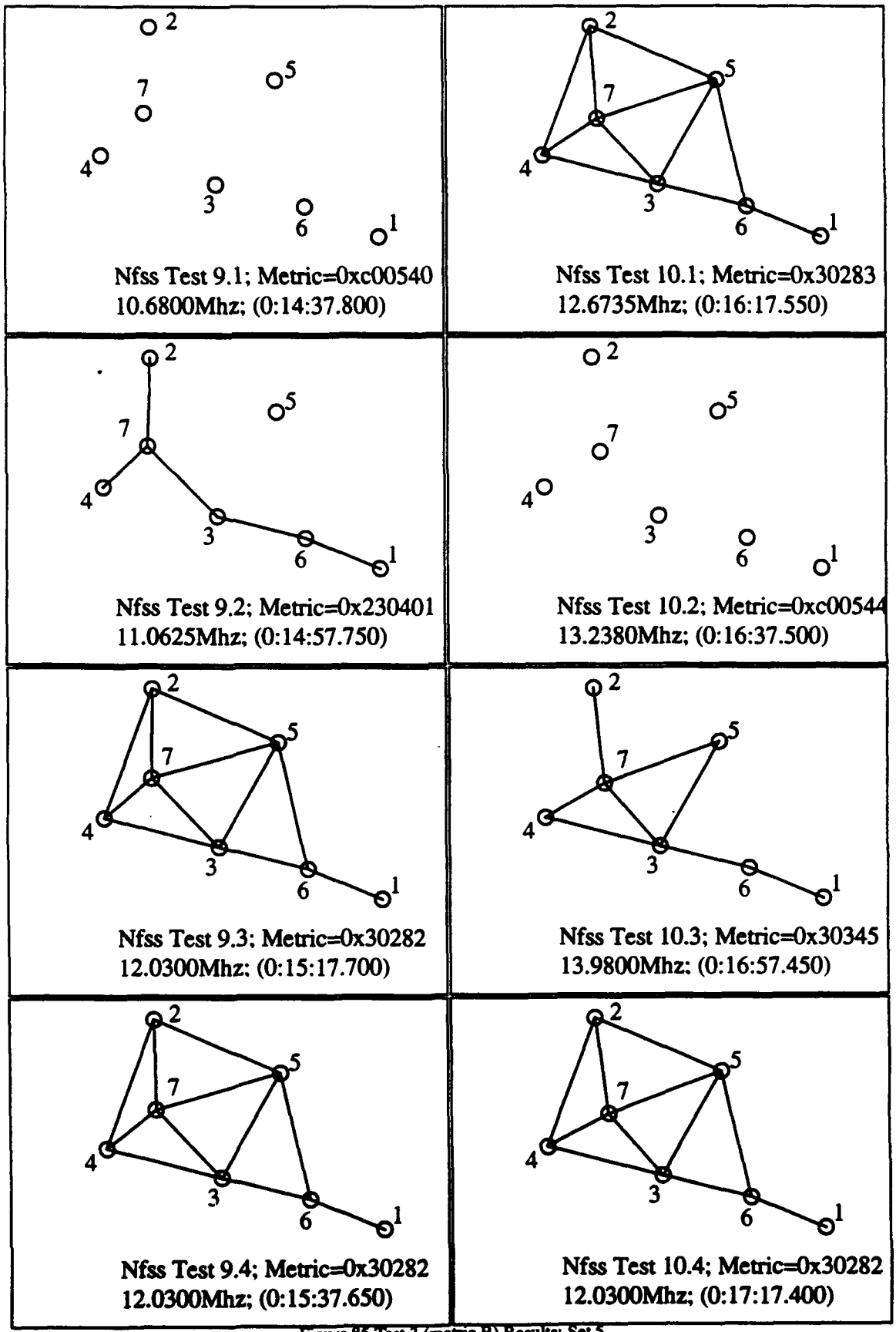


Figure 85 Test 3 (metric B) Results: Set 5

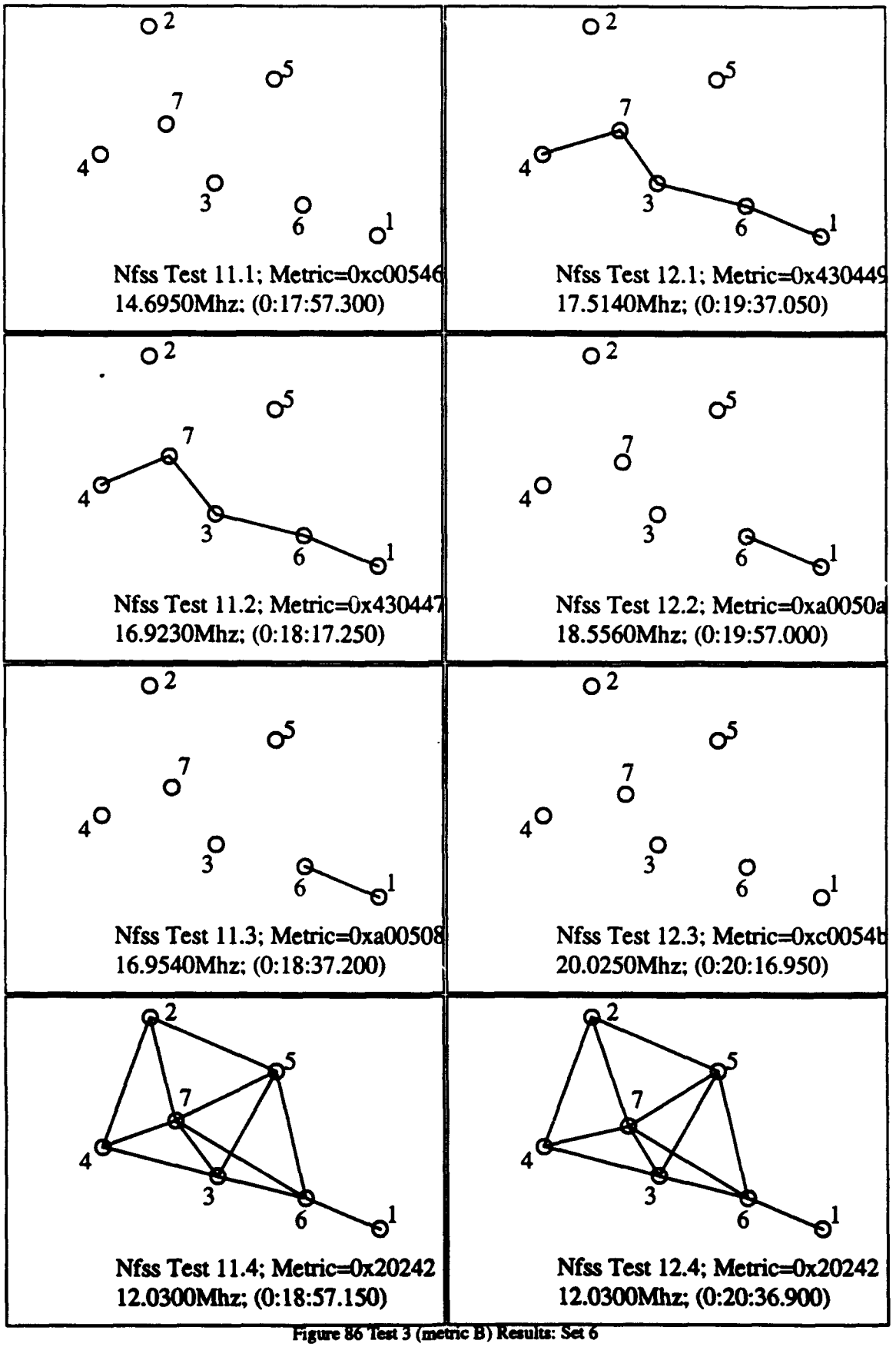


Figure 86 Test 3 (metric B) Results: Set 6